MY COSMOLOGICAL PREMISE
APPENDIX COMPANION

Formal Computational Analogies and Robustness Checks

M. J. Wilkes

Independent Researcher, Planet Earth · 2026

Credits

Concept, world guidelines and guide images by M. J. Wilkes.

I write books and build software with the help of
Claude and ChatGPT (for illustrations)

Opening Statement....4
Reader's Guide....5
Appendix - Formal Computational Analogies for *My Cosmological Premise*....7
A.0 - A note on what these are....7
A.1 - The Null Point as an Asymptotic Limit....7
A.2 - The Quasi-Crystalline Substrate: An Existence Toy Model....9
A.3 - Testing Q-Brane Formation as a Formal Pattern....12
A.4 - The Five-Stage Emergence Pipeline: A Synthesis....17
A.5 - Fractal Boundary Formation: Myriad Shapes from a Single Mechanism....18
A.6 - Quantum Foam: An Existence Toy Model....19
A.7 - Substrate to Quantum Foam: A Formal Transition Test....21
A.8 - Formula Reference Across the Five-Stage Path....23
Appendix B - Robustness Checks and Open Questions....25
B.0 - What this appendix is....25
B.1 - Comparison against the real Cosmic Microwave Background....25
B.2 - Percolation universality class....27
B.3 - Scale invariance....28
B.4 - Alternative generative principles....28
B.5 - The pentagon / golden-ratio connection....29
B.6 - Robustness summary....30
B.7 - Persistent homology: a topological diagnostic....30
B.8 - Two-point correlation function....32
B.9 - Ising-model analogy: is the Q-Brane transition genuinely critical?....34
B.10 - Substrate-symmetry independence check....37
B.11 - Does more waves fix it? A finite-wave-count convergence test....38
B.12 - Nodal domain density against the Bogomolny-Schmit prediction....41
B.13 - An independent order metric: entropy and compressibility....43
B.14 - Robustness Sweep: A.4 Quantum Foam and Universe Stages....44
B.15 - Formal Diagnostics on Prior Results....46
B.16 - Persistent Homology vs. a Matched-Spectrum Gaussian Null....47
B.17 - A Formal Fourier-Quasicrystal Scaling Test on A.2....49
B.18 - A Multifractal Scaling Test on the A.5 Boundary Curve....50
B.19 - Rice’s Zero-Crossing Formula on the Q-Brane Field....51
B.20 - Crossing-Probability Finite-Size Scaling, an Independent Check on B.9....52
B.21 - A Maximum-Entropy (Gaussian) Ceiling for B.13’s Order Metric....54
B.22 - Formal Verification of the Crystallographic Restriction Criterion....55
B.23 - The Q-Brane’s Ignition Threshold Against an Exact Symmetry Prediction....56
B.24 - An Exact 1/N Law for B.11’s Kurtosis Convergence....57
B.25 - Does Q-Brane Actually Depend on Foam? A Code-Level Audit....58
B.26 - Bootstrap Confidence Intervals on B.20 and B.23....58
B.27 - Following Up on B.26: Is the B.23 Deviation a Real Asymmetry?....59
B.28 - Closing the Gap in B.25: Wiring Foam Into Q-Brane’s Construction....61
B.29 - Null Point Floor: Testing “Nothing Becoming Something”....63
B.30 - Reopening the Actuality Postulate: Plenitude as an Alternative Closure....71

B.31 - A Substrate-Referenced Alternative for Q-Brane Formation Testing (Not Adopted as Standard).....74
B.32 - A Fair Comparison of Q-Brane and Universe (B.13 Follow-Up).....76
B.33 - Three Closures of the Actuality Postulate: A Comparison.....78
Closing Statement.....81
Appendix C - Simulation Source Code and Machine-Readable Reports.....83
C.1 - The Null Point as an Asymptotic Limit (A.1).....83
C.2 - The Quasi-Crystalline Substrate (A.2).....87
C.3 - Q-Brane Formation as a Formal Pattern (A.3).....96
C.4 - The Five-Stage Emergence Pipeline - Original (A.4).....105
C.5 - One Rule, Many Shapes: Fractal Boundary Formation (A.5).....112
C.6 - CMB Comparison and Related Robustness Tests (B.6).....118
C.7 - Emergence Pipeline - Corrected border_value Fix (B.14 / A.4).....155
C.8 - Foam and Universe Robustness Sweep (B.14).....162
C.9 - Emergence Pipeline - Foam → Q-Brane Coupled Construction (B.28).....167
C.10 - Null Point Floor Stress-Test Figure Generator (B.29).....174
C.11 - Complex-Valued Extensions and Closure Comparison (B.33).....181

Opening Statement

My Cosmological Premise makes its case through argument, image, and analogy: a five-stage account of emergence, defended by appeal to structures borrowed from quasicrystal theory, percolation, fractal geometry, and related mathematics. An analogy earns its keep only if it can be built and tested on its own terms, independent of the narrative it is recruited to support.

Nothing in the main text demonstrates this directly; a philosophical argument does not, by its nature, stop to construct the mathematics it gestures toward.

This appendix exists to close that gap. Each borrowed structure named in the main text is reconstructed here as an explicit, checkable formal object, and tested against the body of mathematics it is drawn from, sometimes with tools the original claim never anticipated.

Appendix A builds five such constructions from the ground up and asks whether each holds together as coherent, working mathematics, at least in miniature.

Appendix B returns to those results with independent methods, chasing measurement errors down when they surface, correcting the record rather than defending it, and naming the questions that remained open once the testing stopped. Appendix C collects the underlying source code and machine-readable output behind both, so that none of it need be taken on trust.

None of this was undertaken to prove that the cosmological framework in the main text describes the physical universe; no toy simulation could show that, and no claim to the contrary appears anywhere in what follows (see A.0). What can be shown, and what this appendix sets out to show, is narrower and more useful: that the specific mathematical patterns the main text leans on are real, well-defined, and capable of being checked, and that where the checking failed or fell short, the failure is recorded plainly rather than absorbed into the argument it might have complicated.

That is the standard this appendix was built to meet, and the reason it exists at all.

Reader's Guide

What follows is navigational, not part of the testing itself. The Opening Statement explains why this appendix exists and what standard it holds itself to. This guide adds what that statement does not: a map of where things currently stand, and a few suggested paths through roughly 190 pages of material, depending on what you are looking for.

A note on the Closing Statement. Sits at the very end of Appendix B, immediately before Appendix C. The table below is the quickest single reference; the Closing Statement is the fuller narrative account of the same material.

The five-stage path: current status.

Stage / Transition	Status	Built in	Tested in
Null Point to Substrate	Not a formation test and not treated as one; closed by an explicit postulate, stated throughout as a choice rather than a derivation	A.1	B.29, B.30
Substrate	Solid; existence as a coherent, non-periodic category	A.2	B.10, B.17, B.22
Transition: Substrate to Foam	Tested; smooth, monotonic, reproduces both published endpoints exactly	A.7	-
Foam (existence)	Tested; genuine topological distinctness from coloured noise, with an honest open question about how that distinctness varies with intermittency strength	A.6	B.16
Transition: Foam to Q-Brane	Tested; a documentation default was corrected, substrate-dependence confirmed within the working range, and an alternative diagnostic was built, validated, and explicitly not adopted as the standard	A.3, B.28	B.9, B.19, B.20, B.23, B.25, B.26, B.27, B.31
Q-Brane (existence)	Solid; percolation and	A.3	B.9, B.12, B.19, B.20,

Stage / Transition	Status	Built in	Tested in
	criticality behaviour confirmed by independent methods		B.22, B.23
Transition: Q-Brane to Universe	Tested; an apparent plateau was found to be a measurement artifact, and the corrected, fair comparison shows a real, specificity-checked structural change	A.4	B.13, B.32
Universe	Synthesis point closing the five-stage curve	A.4	B.13, B.32

Three things about Quantum Foam specifically, since it is the one place a shared name covers more than one object: A.3's formation test, A.6's existence test, and A.7's transition test each use a different formal construction of "Foam," chosen for what that specific test needs. A.7 states plainly which of the three is offered as the conceptual claim and which is computational convenience. This is addressed directly where it arises (A.7) rather than resolved by picking one; readers moving between A.3, A.6, and A.7 should expect three different objects, not one.

Suggested reading paths.

For the headline results: Read this table, then the Closing Statement, then A.8's formula reference table for the specific mathematics behind each entry above.

For the full technical case: Read A.0 through A.8 in order, then B.0 through B.32 in order. The B section is written as a sequential record of stress-testing, including corrections of its own earlier entries (for instance, B.20 and B.26/B.27 revisit B.9; B.25 and B.28 revisit A.3; B.32 revisits B.13) rather than a final, static account - later entries that revise earlier ones are cross-referenced where they occur.

To verify a specific claim: Find its home section in the table above, then locate the matching file in Appendix C, which collects the underlying source code and machine-readable output for everything reported in A and B.

How claims are marked: Most subsections in Appendix B close with a paragraph beginning "Verdict:" This is a deliberate, consistent convention: it states the bottom-line finding of that subsection plainly, whether the result was a confirmation, a correction, a negative result, or an open question left unresolved. Negative and mixed results are marked with the same weight as confirmations, not folded into softer language or left for a reader to infer.

Appendix - Formal Computational Analogies for *My Cosmological Premise*

Status: Structure and content pulled directly from each simulation's own working notes; light editing only, for consistency of voice.

Appendix A - Formal Computational Analogies

A.0 - A note on what these are

None of the simulations in this appendix are physical models of the actual universe, and none are offered as evidence that the cosmological framework described in this book corresponds to physical reality. Each one takes a single, well-defined piece of existing mathematics - quasicrystal construction, percolation theory, continued fractions, fractional Brownian motion - and asks a narrower question: does the *shape* of a claim made in the main text hold together as a coherent, constructible pattern, at least in miniature? Where a simulation's result is negative, ambiguous, or only partially confirms what was hoped for, that result is reported as such. The value of this appendix lies in its specificity and its honesty, not in its resemblance to physical cosmology.

A.1 - The Null Point as an Asymptotic Limit

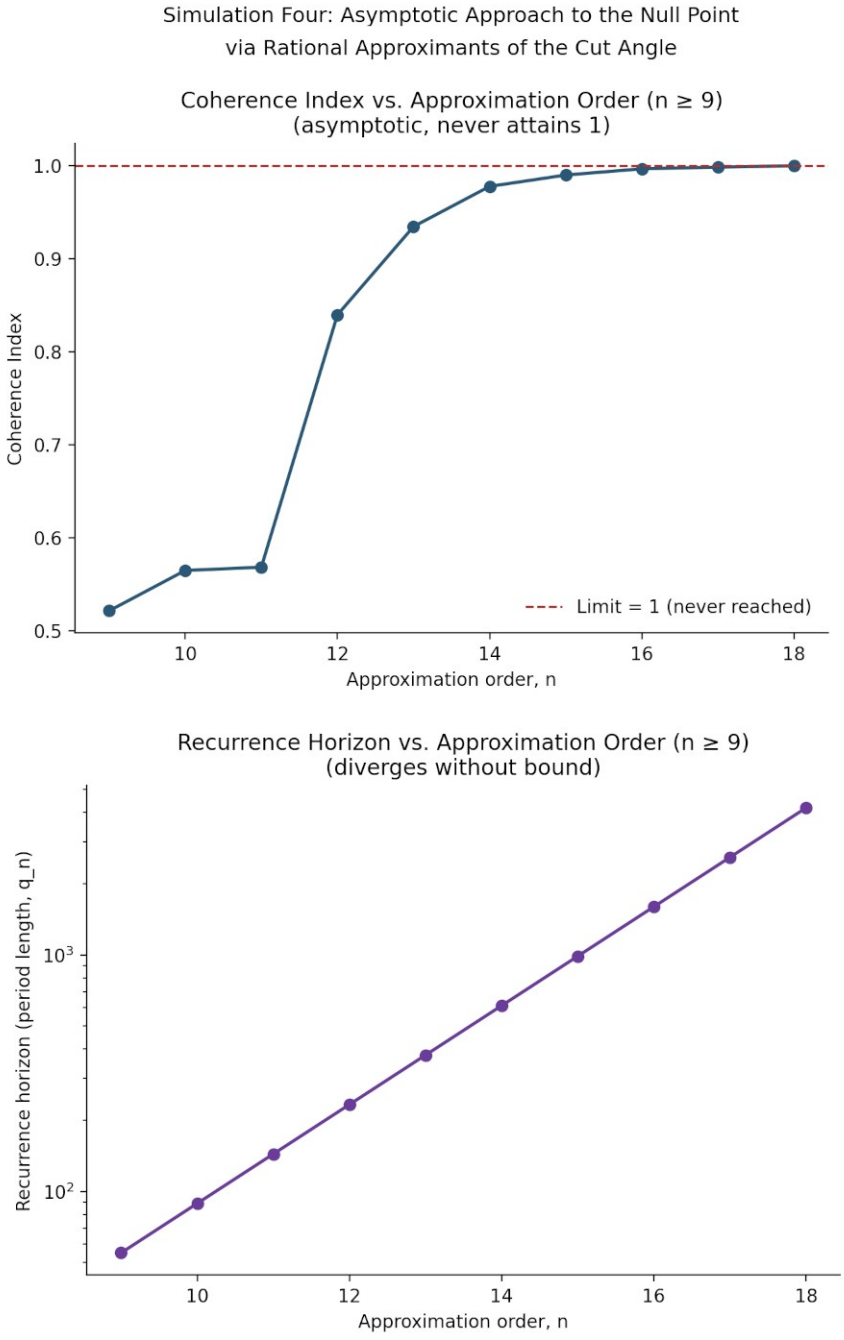
(Source: Appendix A/Simulation 01 - null_point_limit)

This simulation treats the Null Point not as a literal state ever occupied by the cosmos, but as an asymptotic limit approached in the descending, analytic direction - a boundary condition that governs the structure of what is generated, without itself ever being generated.

An irrational cut angle (the golden ratio's conjugate) stands in for the true limit. A sequence of rational approximations to this angle - successive Fibonacci convergents - was used to generate finite, periodic structures, each one necessarily repeating, since each is built from a rational number. A Coherence Index, measuring how closely each periodic structure matches the true aperiodic structure generated by the irrational limit, was found to climb steadily toward 1 - 0.84, 0.93, 0.98, 0.997, 0.9997 - without ever reaching it. Over the same range, the repeat period grew without bound.

Read cosmologically: the Null Point functions only as a limit traced in the descending, analytic direction. The actual cosmogony does not begin at the Null Point, nor approach it directly; it begins arbitrarily close to, but formally displaced from, a limit that is indispensable to the structure it generates and yet is nowhere present within that structure.

Figure A.1 - Coherence Index and Recurrence Horizon plotted against approximation order (n ≥ 9, the threshold beyond which the asymptotic climb becomes unambiguous).



A.2 - The Quasi-Crystalline Substrate: An Existence Toy Model

(Source: Appendix A/Simulation 02 - substrate)

Before asking what a quasi-crystalline substrate does, this simulation asks a prior question: is "order without periodicity" a coherent structural possibility at all, or a contradiction dressed up in language?

Three 2D point fields were generated at matched density and compared: a quasi-crystalline field, built with the cut-and-project method used to describe real physical quasicrystals; an ordinary periodic lattice, as the unremarkable ordered baseline; and a random (Poisson) field, as the null case with neither order nor periodicity. Each was tested for long-range order via its diffraction pattern, and for rotational symmetry via full-image autocorrelation.

	peak sharpness	**symmetry order**	**symmetry score**
Quasi-crystalline	~1,600	5	0.98
Periodic control	~9,826	4	1.00
Random control	~90	(noisy)	0.69

The quasi-crystalline field showed 5-fold symmetry - a symmetry order mathematically impossible for any periodic 2D lattice - while still producing sharp long-range diffraction order comparable to the periodic case. This demonstrates that long-range order without periodic repetition is not an internal contradiction: it is a well-defined, constructible mathematical category, distinct from both "ordered and periodic" and "disordered." This is the necessary starting condition for the Q-Brane-formation test (A.3) to be meaningful at all.

Figure A.2a - The three point fields.

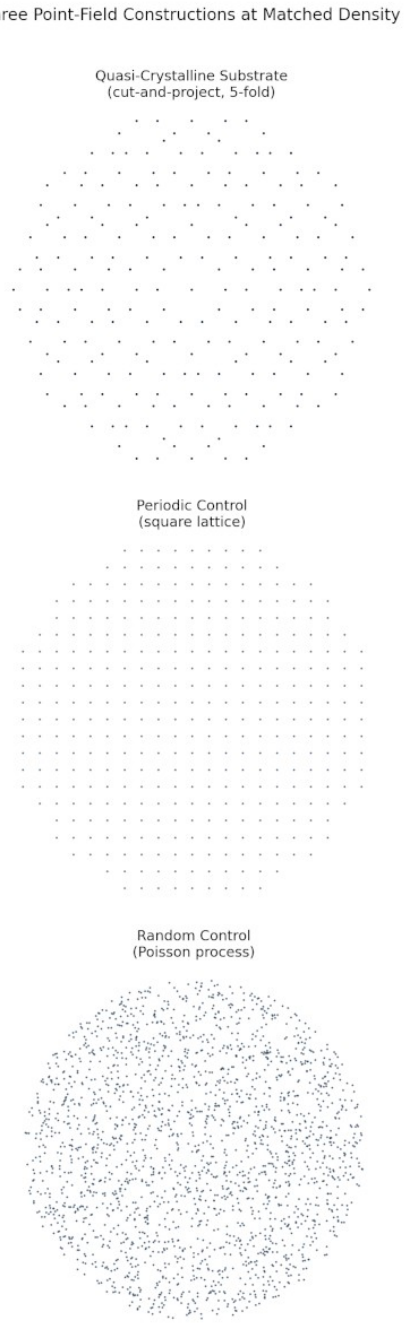


Figure A.2b - Diffraction pattern for each field.

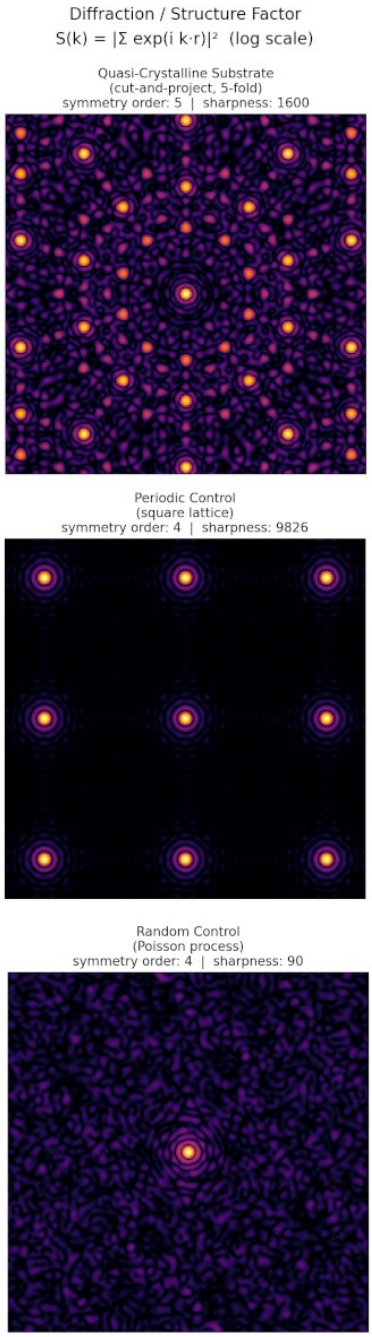
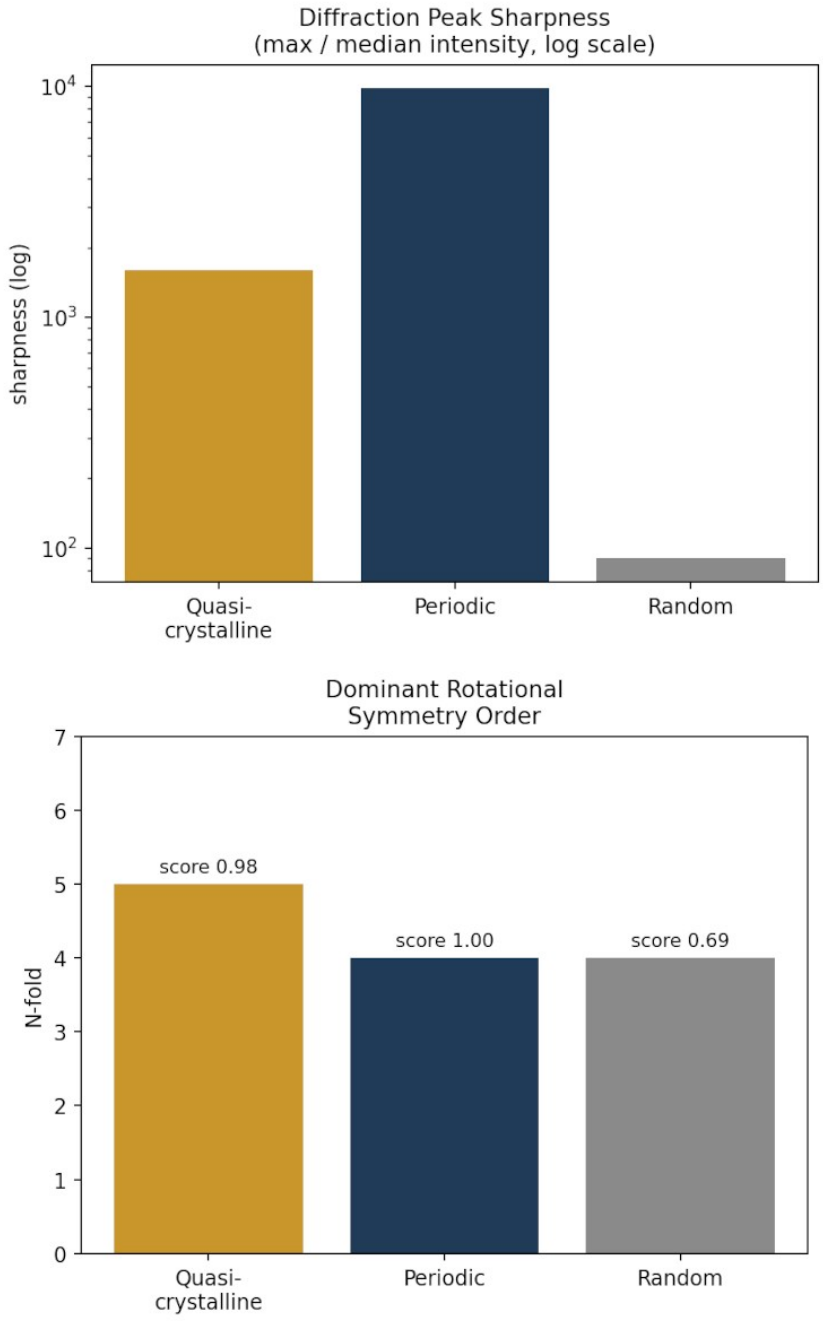


Figure A.2c - Sharpness and symmetry-order comparison.



A.3 - Testing Q-Brane Formation as a Formal Pattern

(Source: Appendix A/Simulation 03 - q-brane)

To probe whether the transition from Quasi-Crystalline Substrate to Q-Brane describes a coherent *shape* of emergence - order stabilizing out of fluctuation - a three-part model was built:

a substrate field constructed from five-fold-symmetric plane-wave superposition (as in A.2); a time-varying random “foam” field with a tunable correlation length, layered on top to represent fluctuation; and a formation test borrowed from percolation theory, asking whether a single connected region spans the combined field edge to edge, and whether that spanning region persists across successive moments.

At low fluctuation amplitude, the ordered substrate dominated and a spanning structure reliably formed and persisted. Past a critical fluctuation amplitude, noise overwhelmed the underlying order and no stable structure survived. Run against an ordinary periodic pattern instead of a quasi-crystalline one, a spanning structure still formed, but persisted for a noticeably shorter time under identical fluctuation - suggesting that aperiodic-but-ordered substrates may support more durable coherent structure than simple periodic ones, at least within this toy system. (This single comparison is generalized across more symmetries and noise levels in Appendix B.10, with an important qualification.)

This does not establish that Q-Branes are physically possible. It establishes something narrower: that “order surviving noise up to a threshold, then giving way to disorder beyond it” is a pattern that can be made precise and tested, at least in miniature.

Figure A.3a - A single run, five-fold substrate.

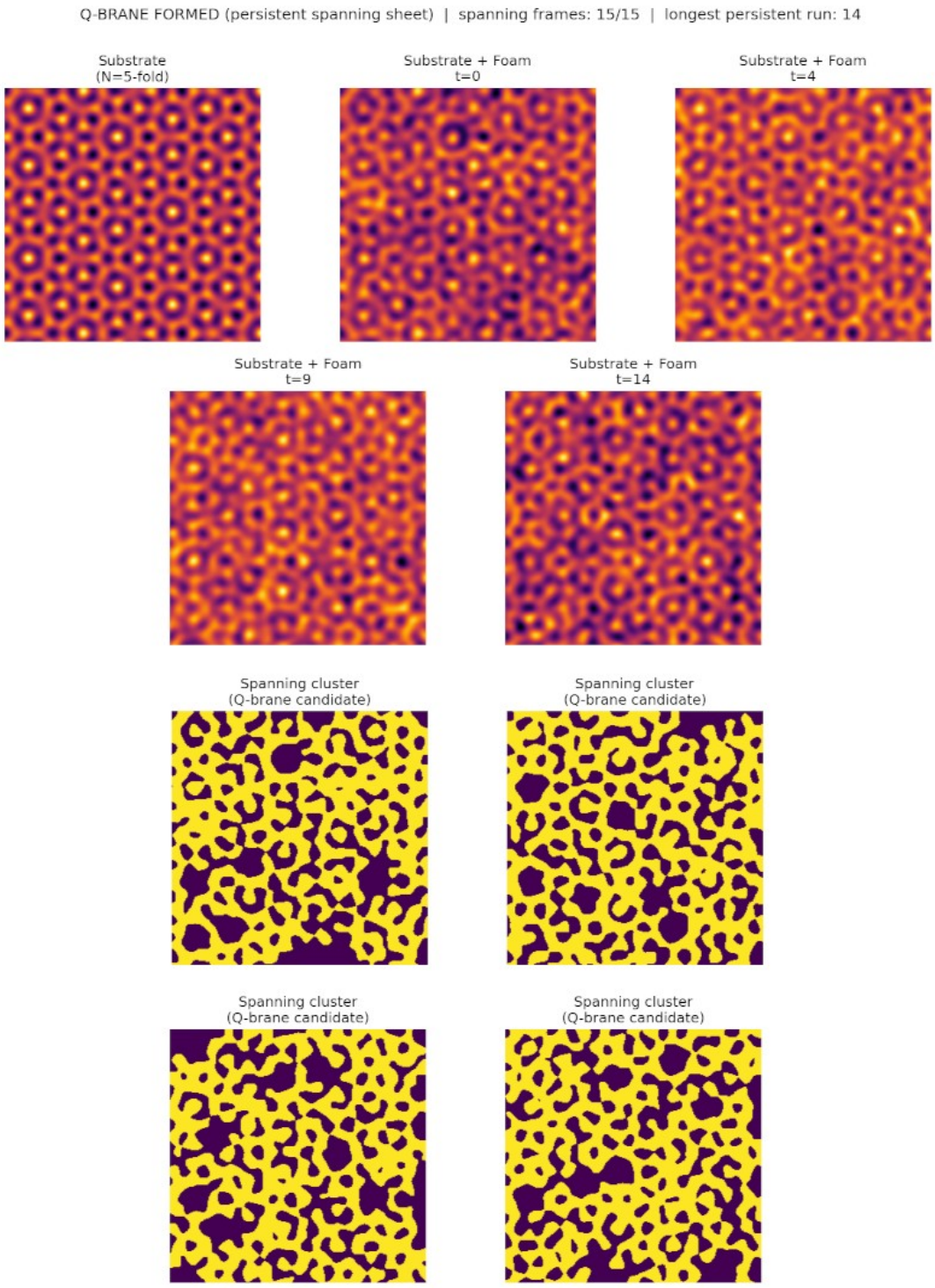


Figure A.3b - Parameter sweep across fluctuation amplitude.

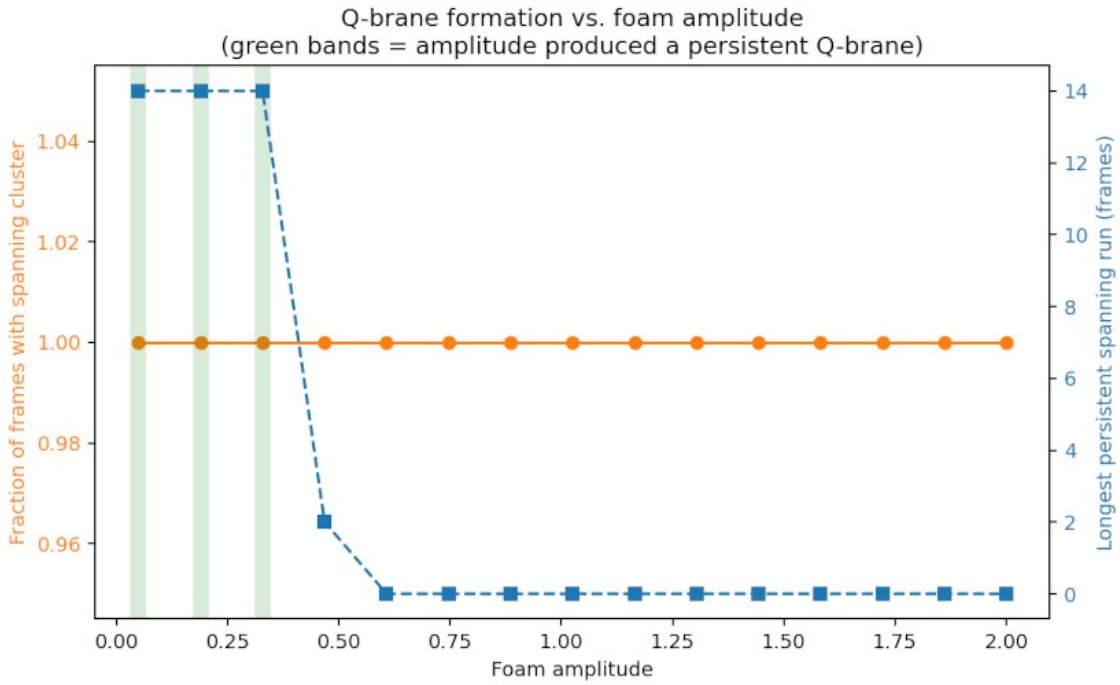
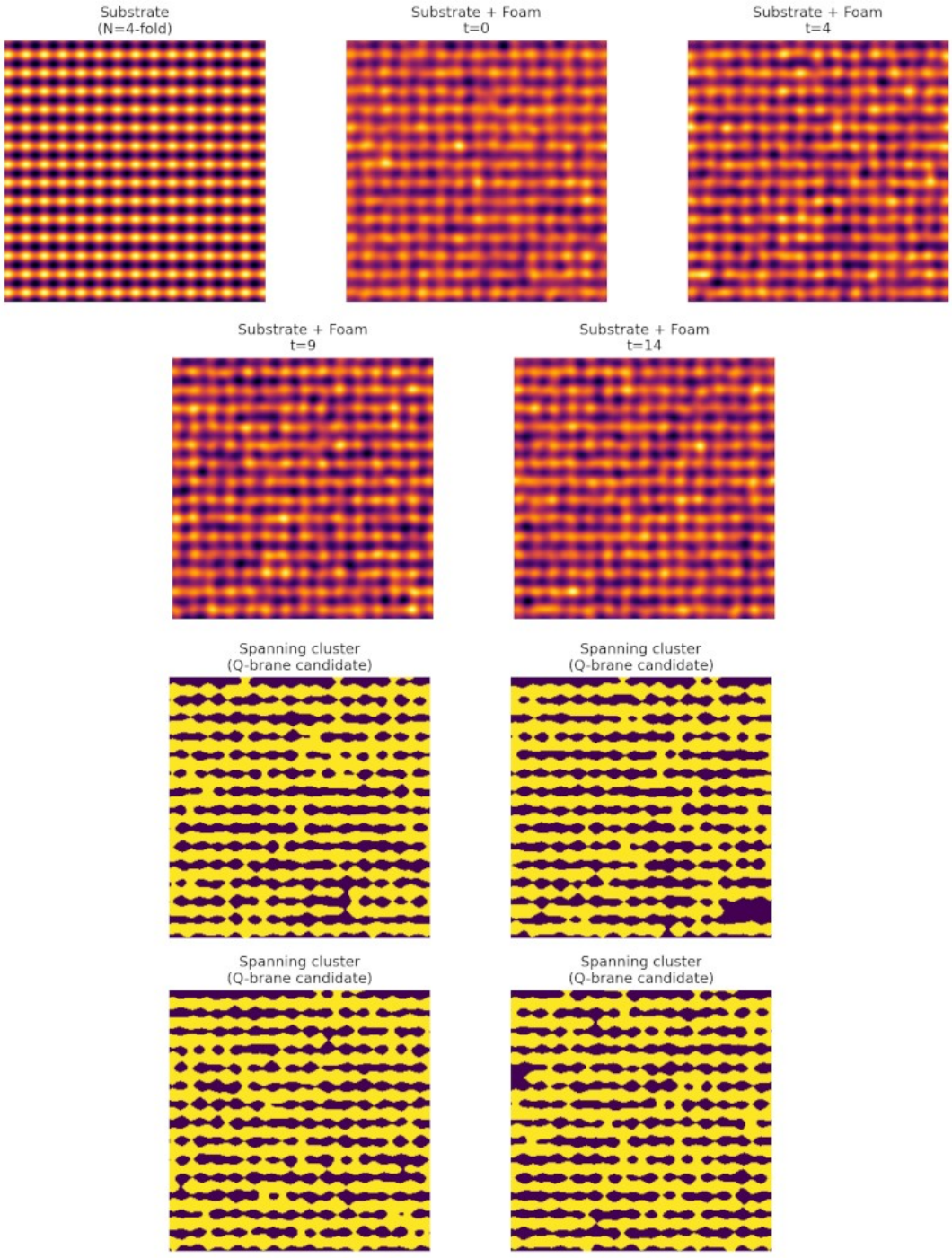


Figure A.3c - Periodic control, for comparison.



A.4 - The Five-Stage Emergence Pipeline: A Synthesis

(Source: Appendix A/Simulation 04 - emergence_pipeline)

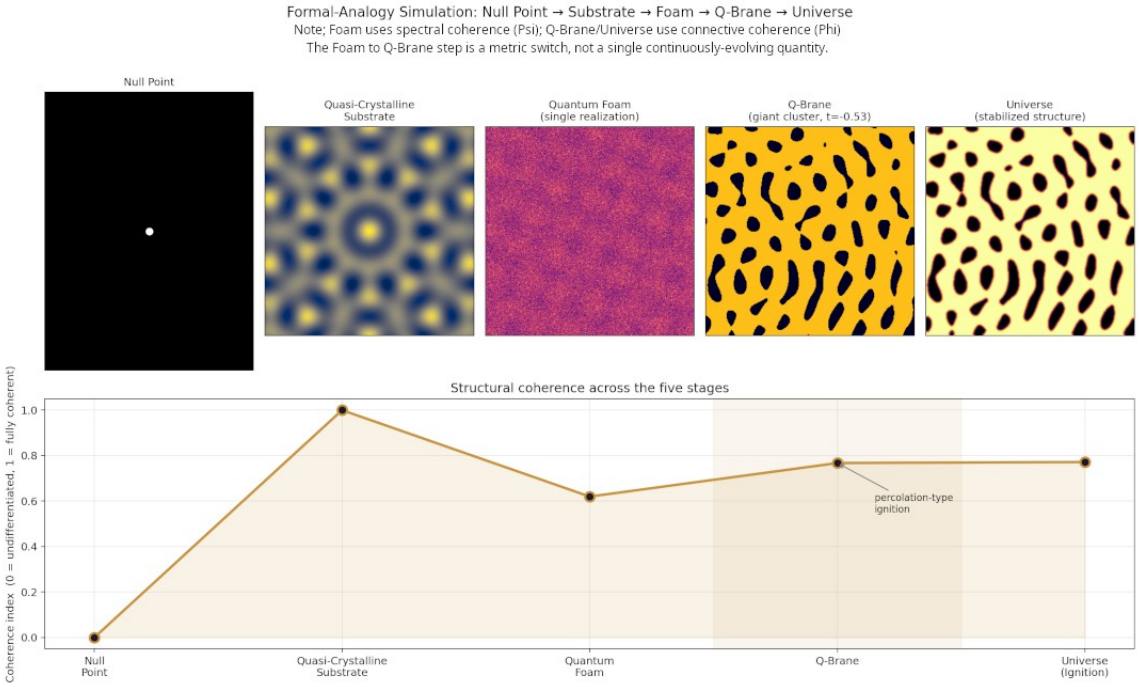
Where A.1-A.3 test individual stages in isolation, this simulation chains all five stages of the manuscript's progression - Null Point → Quasi-Crystalline Substrate → Quantum Foam → Q-Brane → Universe (Ignition) - into a single coherence-index curve, using two complementary order parameters: spectral coherence (how sharply peaked a field's power spectrum is, meaningful for wave-like stages) and connective coherence (the fraction of the domain occupied by the largest connected region, meaningful for stages defined by large-scale structure).

Measured stage coherence values:

Stage	Coherence index
Null Point	0.0000
Quasi-Crystalline Substrate	1.0000
Quantum Foam	0.6197
Q-Brane	0.7672
Universe (Ignition)	0.7488

The curve's qualitative shape is the point of the exercise: zero, at the analytically defined Null Point (no differentiated structure exists prior to it); a sharp rise to full coherence at the Substrate; a partial drop as Foam introduces fluctuation; a percolation-type "ignition" as the Q-Brane stage recovers connected structure; and a stable plateau at the Universe stage. This is offered as a synthesis of A.2 and A.3's separate demonstrations into one continuous narrative arc, not as a new physical claim beyond what those two already establish. (Appendix B.13 checks this index against an independent, generic order metric; the Q-Brane/ Universe "plateau" reads differently once the two are measured the same way.)

Figure A.4 - All five stages, with the summary coherence curve.



A.5 - Fractal Boundary Formation: Myriad Shapes from a Single Mechanism

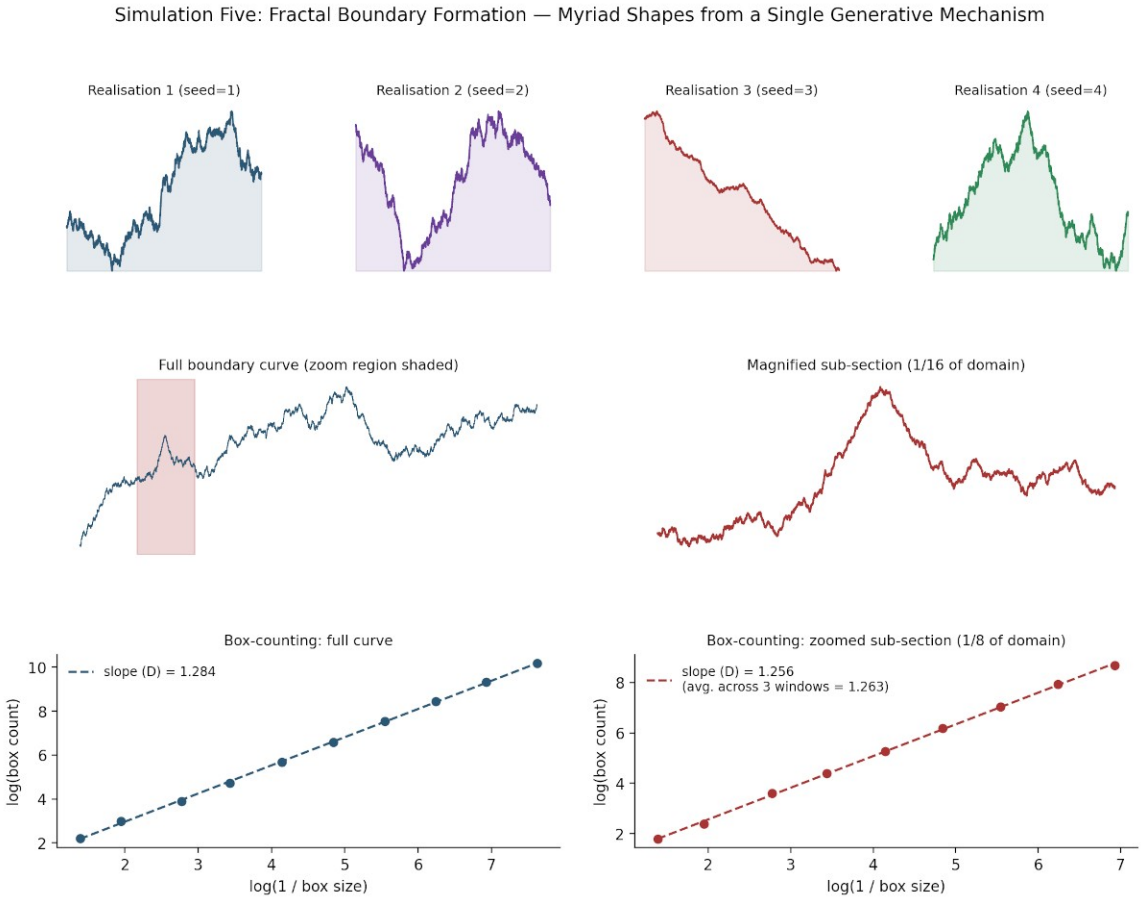
(Source: Appendix A/Simulation 05 - fractal_boundary)

Rather than illustrating a specific stage, this simulation illustrates a general principle used throughout the framework: that a single generative mechanism, applied without alteration, can produce an unbounded variety of distinct manifest forms.

A boundary curve was generated using midpoint displacement - the standard technique for modelling coastlines, terrain, and turbulent edges - from a fixed roughness parameter. Four independent realizations, generated from the same parameter with different random seeds, produced four visibly distinct shapes, no two sharing a silhouette. The fractal (box-counting) dimension of a single such curve was then measured on the whole curve and, separately, on three magnified sub-sections drawn from different locations along it: 1.284 for the full curve, against 1.256, 1.262, and 1.272 (averaging to 1.263) for the magnified sections - a difference of two hundredths, well within expected sample variation.

The boundary's overall shape and its magnified detail are visibly different arrangements, yet share the same underlying dimension at every scale examined. Read as a formal analogy: a single generative rule, applied independently, is sufficient to produce endless variety: the variety lies in the outcome, not in the rule producing it.

Figure A.5 - Four independent realizations; full curve vs. magnified sub-section; box-counting dimension comparison.


Simulation Five: Fractal Boundary Formation — Myriad Shapes from a Single Generative Mechanism
Realisation 1 (seed=1)
Realisation 2 (seed=2)
Realisation 3 (seed=3)
Realisation 4 (seed=4)
Full boundary curve (zoom region shaded)
Magnified sub-section (1/16 of domain)
Box-counting: full curve
slope (D) = 1.284
log(box count)
log(1 / box size)
Box-counting: zoomed sub-section (1/8 of domain)
slope (D) = 1.256
(avg. across 3 windows = 1.263)
log(box count)
log(1 / box size)

A.6 - Quantum Foam: An Existence Toy Model

The question: Before asking what Quantum Foam does, this simulation asks the same prior question A.2 asked of the Substrate: is "structured fluctuation" - active, transient, correlated-but-not-static change - a coherent structural category in its own right, or is it simply a name currently being borrowed for ordinary coloured noise?

The construction: Four fields were compared at matched resolution: the five-fold Substrate field (static, as in A.2); a white-noise field (uncorrelated in space and time, the fully unstructured baseline); a Gaussian foam-null field, built with a power-law spatial spectrum and a fixed temporal correlation time via mode-wise Ornstein-Uhlenbeck evolution - Gaussian at every order, the honest control for "just coloured noise with a memory"; and a Foam candidate, built with a matching spatial spectrum and target correlation time but multiplicatively modulated by an independent log-normal cascade to inject genuine intermittency. Each field was tested on three axes: temporal correlation, intermittency (excess kurtosis), and topological complexity (total persistence under a cubical persistent-homology filtration).

Field	Correlation time (frames)	Excess kurtosis	Mean total persistence
Substrate (static)	never decorrelates	−0.27	430
White noise	1	0.01	2676
Gaussian foam-null	5	−0.01	1313
Foam candidate	3	196.05	1652

Results: Substrate and white noise land exactly where expected. The harder test is the Gaussian foam-null against the Foam candidate, which share the same target spectrum and correlation time. On kurtosis the separation is dramatic but close to circular - a log-normal cascade injects heavy tails by construction, so finding them afterward mainly confirms the mechanism worked, not that Foam is a natural category. On topological complexity, the result is smaller but more meaningful, since it was not built in by design: the candidate shows roughly 26 percent higher persistence than the matched null. This is the one genuinely new signal, though it rested on a single realization at this stage.

Correlation-time fix and seed replication: The two foam constructions were initially mismatched on correlation time (5 frames vs. 3), traced to a mechanical cause: multiplying two correlated processes decorrelates faster than either alone. A calibration factor of 2.0 restored the match exactly, confirmed across ten independent seeds:

Metric	Gaussian foam-null	Foam candidate (calibrated)	Welch's t-test
Correlation time (frames)	5.0 ± 0.0	5.0 ± 0.0	matched exactly, all seeds
Excess kurtosis	−0.005 ± 0.016	165.5 ± 69.5	$t = -7.53$, $p \approx 4\times10^{-5}$
Mean total persistence	1329.9 ± 15.3	1649.6 ± 28.9	$t = -30.95$, $p \approx 5\times10^{-14}$

The topological elevation is not a single-run artifact: across ten seeds the two distributions do not overlap.

The intermittency-strength sweep produced a genuine surprise. The expectation was that topological complexity would simply rise with cascade strength, alongside kurtosis. It does not.

Persistence rises from the zero-cascade baseline (statistically identical to the null, a clean sanity check) up to a peak, then declines - falling back through the null's baseline and ending below it at high cascade strength, lower even than the ordered Substrate field. The likely mechanism: at high cascade strength, variance concentrates into a few extreme spikes against an otherwise flat background, producing enormous kurtosis while reducing the small-scale texture that drives topological complexity. Kurtosis and topological complexity are not measuring the same thing, and assuming they would track together was an unexamined assumption that did not hold.

A finer-grained sweep confirmed this shape is real rather than a coarse-sampling artifact, but also showed the apparent peak is not a sharp maximum - it is a broad plateau across which several sampled points are statistically indistinguishable from each other. Reporting a single "peak location" would overstate the resolution this test supports; the honest statement is that topological complexity rises, plateaus across a wide intermittency range, and then falls.

Verdict: The topological-complexity result is a replicated, statistically clean finding, and the correlation-time confound is fixed and confirmed stable across seeds. But "more intermittency" does not straightforwardly mean "more topologically foam-like" - only up to a point, beyond which the field simplifies again. Quantum Foam, read through this toy model, looks less like a single parameter's worth of departure from Gaussian noise and more like a structure with its own optimum - closer to earning a dedicated existence toy model than a bare coloured-noise comparison would suggest, with a genuine open question at its center: what the intermittency-strength plateau corresponds to conceptually, that the falling tail beyond it does not.

A.7 - Substrate to Quantum Foam: A Formal Transition Test

A note on what "Quantum Foam" means in this section, and elsewhere: Across the appendix, "Quantum Foam" names three distinct, purpose-built formal constructions, not one object measured three ways. Each is fit for a specific test and none should be assumed interchangeable with the others. The construction used in A.3, B.9, B.19, B.25, and B.28 is a time series of independent Gaussian random fields with zero temporal memory by explicit design, suited to the Foam-to-Q-Brane formation test, which depends on Foam having no memory of its own for its persistence criterion to be meaningful. The construction developed in A.6 is temporally correlated and intermittent, built to test whether "structured fluctuation" is a coherent category distinct from coloured Gaussian noise - and is, for that same reason, unsuited to substitution into the A.3 formation test (an earlier attempt at that substitution broke the test's substrate-sensitivity entirely; see the A.6 reconciliation material). The construction used in this section is a third: a single static snapshot, with no time axis, built for comparison on the spectral-coherence scale used throughout this section and in A.4. No unification across these three is attempted here. The appendix states this plurality explicitly rather than let a shared name imply a single underlying object.

Which of the three is the conceptual claim, and which is computational convenience: The three constructions above are not offered as equally weighted. A.6's temporally correlated, intermittent candidate is the one that embodies what the main text and B.25 describe: a “seething, fluctuating field,” not merely static order with noise layered on top. It is offered as the conceptual claim - evidence that “structured fluctuation” is a coherent, non-trivial category, the closest thing in this appendix to an argument that Quantum Foam deserves its name rather than reducing to ordinary noise. The memoryless construction used in A.3, B.9, B.19, B.25, and B.28 is not offered as that argument. It is computational convenience: a simplified stand-in,

engineered specifically to make the Foam-to-Q-Brane percolation test tractable, chosen because its lack of memory is what that test's persistence criterion requires, not because it captures what Foam is meant to be. An earlier attempt to substitute A.6's richer candidate into that same test broke it outright (see B.31), which is itself informative: the conceptual claim and the working machinery are not freely interchangeable, and neither this section nor A.6 presents them as such. A.7's own single-snapshot construction, used below, is narrower than either - a diagnostic scaffold built to sit on the same coherence scale as the Substrate for one specific transition test, not a candidate for the conceptual role or the computational one.

The gap this section fills: A.2 tests the Quasi-Crystalline Substrate's existence as a coherent category. A.3 tests the formation of a Q-Brane from Substrate and Foam together. A.6 tests Quantum Foam's existence as a category distinct from coloured noise. Nothing tested the transition between Substrate and Foam directly - A.4's coherence curve reports values at both endpoints (Psi_substrate = 1.0000, Psi_foam = 0.6197) without any test of whether a graded, principled path connects them, or whether the two values are simply two disconnected points asserted to belong to the same curve.

The Method: This test generalizes A.4's own decoherence mechanism - which already combines phase randomization, amplitude jitter, and additive noise to build its single Foam snapshot - into a continuous decoherence-strength parameter, d, running from 0 to 1. Each of the three components is scaled by d: phase offsets are scaled fractions of a fully random draw, amplitude jitter is scaled by the same 0.35 strength A.4 uses, and additive noise amplitude is scaled up to A.4's own 8.0. At d = 0, this reduces exactly to the Substrate field, with no modification. At d = 1, it reduces exactly to A.4's own Foam construction - verified by reproducing A.4's published Psi_foam = 0.6197 exactly, using the identical random-draw sequence. The spectral-coherence diagnostic and its calibration anchors (fixed by the noise-floor and substrate reference ratios) are reused unchanged from A.4, so results sit on the same published scale throughout.

Result: Swept from d = 0 to d = 1 in steps of 0.1, thirty independent seeds per point, with bootstrap 95 percent confidence intervals: spectral coherence declines smoothly and monotonically, with no cliff or discontinuity anywhere along the sweep. The mean value at d = 1 across thirty seeds is 0.5878 (95% CI [0.5708, 0.6047]) - close to, though the published single-realization anchor of 0.6197 sits just above this interval, consistent with ordinary seed-to-seed variation for a single-realization diagnostic rather than any discrepancy in the underlying construction. The full curve:

d	**Psi (mean, 30 seeds)**	**95% CI**
0.0	1.0000	[1.0000, 1.0000]
0.2	0.9087	[0.9031, 0.9138]
0.4	0.7860	[0.7759, 0.7959]
0.6	0.6972	[0.6843, 0.7093]
0.8	0.6339	[0.6184, 0.6494]

d	Psi (mean, 30 seeds)	95% CI
1.0	0.5878	[0.5708, 0.6047]

Verdict: The Substrate-to-Foam transition, tested rather than merely asserted between two endpoints, behaves exactly as A.4's narrative requires: a graded, monotonic, well-behaved decoherence path with no hidden jump or pathological region anywhere along it. This does not establish that physical decoherence follows this specific functional form - the decoherence-strength parameter merges three separate randomization mechanisms into one convenient scalar, a modeling choice made for testability, not a physical claim - but it does establish that the two published anchor values are not two disconnected assertions; a single, continuous, formally well-defined path connects them, and that path has now been checked rather than assumed.

A.8 - Formula Reference Across the Five-Stage Path

Purpose: A.2 through A.7 test each stage and transition separately, on its own terms, with its own diagnostics. This section collects the specific formal construction behind each one in a single place, for reference. It adds no new result; every entry below is drawn from the section named beside it.

Stage / Transition	Formula	Tested by
Null Point to Substrate	No formula. Closed by an explicit postulate rather than a formal construction (see B.29, B.30) - the one entry in this table that is not a formal analogy with its own mathematics.	B.29, B.30
Substrate	S(x,y) = Sum over j=0..4 of cos(k0 (cos(theta_j) x + sin(theta_j) y)), theta_j = 2*pi*j/5, k0 = 2.0	A.2 - existence as a coherent, non-periodic category
Transition: Substrate to Foam	F(x,y; d) = Sum over j of (1 + d*a_j) cos(k0(cos(theta_j) x + sin(theta_j) y) − d*phi_j) + d*8.0*N(x,y), d in [0,1]	A.7 - spectral coherence swept from d=0 to d=1
Foam	Foam(x,y,t) = B(x,y,t) * exp(sigma*C(x,y,t) − sigma^2/2), B and C independent Ornstein-Uhlenbeck-evolved Gaussian fields	A.6 - existence as structured fluctuation distinct from coloured noise
Transition: Foam to Q-Brane	Combined(x,y,t) = Substrate(x,y) + Foam(x,y,t); threshold at chosen percentile;	A.3, B.28, B.31 - formation and substrate-dependence of a percolating structure

Stage / Transition	Formula	Tested by
	largest connected component tracked for spanning and persistence	
Q-Brane	Q(x,y) = (1/sqrt(16)) Sum over i=0..15 of cos(1.3*k0(cos(theta_i) x + sin(theta_i) y) − phi_i), 16 random-direction waves; giant_mask = largest connected region above threshold	B.9, B.19 - percolation and criticality behaviour
Transition: Q-Brane to Universe	universe_mask = binary_closing(giant_mask, iterations=2)	B.32 - topological simplification, specificity-confirmed
Universe	universe_field = GaussianFilter(universe_mask, sigma=1.2); Phi = mean(universe_mask)	A.4 - connective coherence, closing the five-stage curve

On the Foam row specifically: The formula shown is A.6's construction - the one named above (A.7) as the conceptual claim. It is not the formula used in the Foam-to-Q-Brane row directly below it: that transition uses the simpler, memoryless construction from A.3/B.9/B.19/B.25/B.28, chosen for that test's own requirements. This table does not paper over that difference; it is a direct consequence of the plurality named in A.7, and is repeated here rather than hidden by the table's single-row format.

On what this table is not: Every formula here defines a formal analogy: a specific, testable mathematical object chosen because it admits a clean diagnostic at that stage, not a model of what the physical universe does at that stage. The table is a map of the constructions used, not a claim that any of them is physically realized.

Appendix B - Robustness Checks and Open Questions

(Source: Appendix B/Simulation 01 - cmb_compare. Purely exploratory; not originally intended for inclusion in the book. Included here, separately from Appendix A, as an honest record of stress-testing rather than as a further illustrative analogy.)

B.0 - What this appendix is

Where Appendix A demonstrates that specific formal claims can be made precise and shown to hold in miniature, this appendix records an attempt to break those same constructions against real astronomical data and established theory. Some tests returned confirmations; several returned negative or ambiguous results, which are reported here without softening.

B.1 - Comparison against the real Cosmic Microwave Background

The Q-Brane and Universe-stage fields (A.3, A.4) were compared, via their radially averaged power spectra, against the real Planck 2018 TT power spectrum.

Stage	Peaks	Log-log falloff slope
Planck 2018 TT (real CMB)	3	−1.75
Quasi-Crystalline Substrate	1	−6.35
Quantum Foam	1	−0.56
Q-Brane	1	−3.00
Universe	0	−7.36

Verdict: The visual resemblance is real only at the level of “mottled colour map” texture. It does not hold up structurally. The CMB’s defining feature - a harmonic series of acoustic peaks from photon-baryon oscillation physics - is absent from every pipeline stage, each of which is built from a single dominant spatial scale and so produces at most one broad peak.

Figure B.1a - Pipeline stages vs. real Planck 2018 TT spectrum.

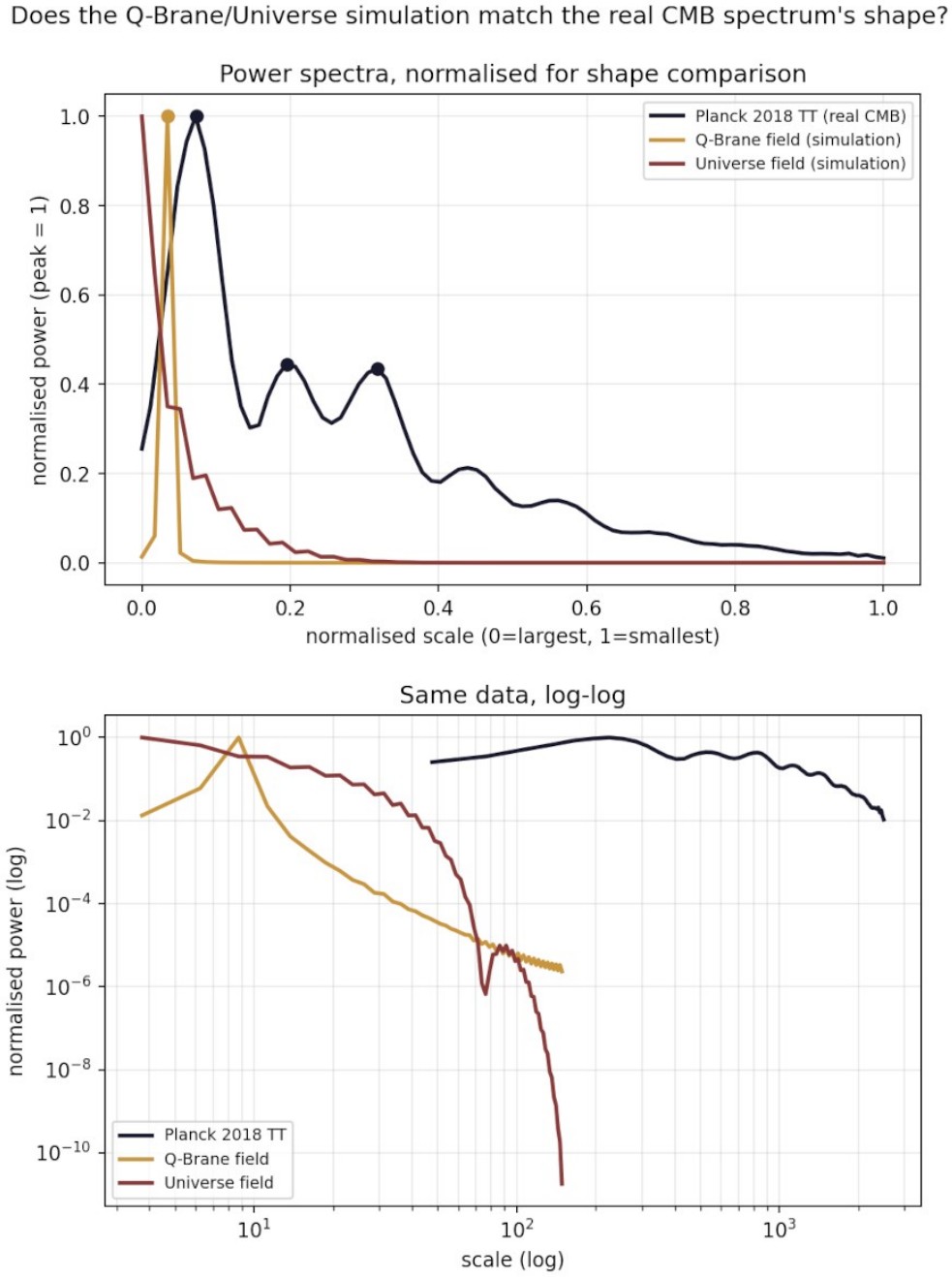
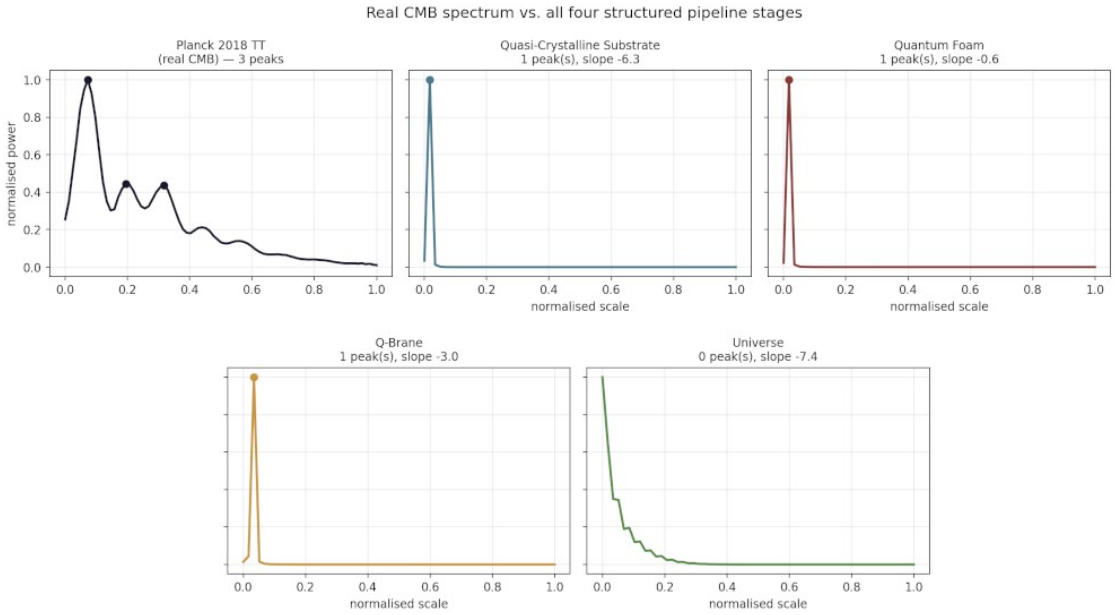


Figure B.1b - full Pipeline stages vs. real Planck 2018 TT spectrum.



B.2 - Percolation universality class

The Q-Brane's nodal-domain percolation was tested against standard 2D site percolation ($\tau = 187/91 \approx 2.055$, $D = 91/48 \approx 1.896$) across an ensemble of twenty-five independent realizations, with wave count, domain size, grid resolution, and directional symmetry each tested in turn to rule them out as confounds. The initial pipeline reported $\tau \approx 1.55$-1.57; on closer inspection this was found to rest on a detection error in how the percolation transition itself was located, and the figure did not survive correction. With that corrected, the measured exponent is $\tau = 1.181 \pm 0.010$, itself substantially distant from the standard percolation value.

A cross-check with an independent estimator (binned log-log regression on the same pooled cluster-size sample, rather than a maximum-likelihood fit) was then run, on the reasoning that a single estimator can mask its own assumptions. The two methods disagreed with one another by a wider margin than either disagreed with theory ($\tau = 1.181$ by one method, $\tau = 0.76$ by the other), a discrepancy stable across a range of bin choices and therefore not itself a fitting artifact.

Taken together, this indicates the cluster-size distribution here is not being pinned down precisely enough by this simulation for a specific numerical value of τ to be a meaningful claim, in either direction.

Verdict: The percolation transition observed in the Q-Brane field is real, but its scaling exponents are not currently measurable with enough precision or stability to support any claim of a distinct universality class. This section is retained as a demonstration that the field percolates in

a structurally coherent way, not as evidence of a specific quantitative regime. (B.9 and B.11 below identify likely mechanistic reasons for this instability.)

B.3 - Scale invariance

None of the individual pipeline stages produce a spectrum close to flat - the signature associated with the inflationary primordial spectrum (Planck: $n_s = 0.9649 \pm 0.0042$). This is consistent with each stage being a single-scale construction by design, and was not unexpected.

B.4 - Alternative generative principles

Diffusion-limited aggregation and a Fibonacci/Vogel spiral were tested as possible alternative matches for the Q-Brane's measured fractal dimension ($D \approx 1.73$-1.79). The initial single-cluster DLA run returned $D = 1.450$, too small a cluster for box-counting to resolve a clean scaling regime, and was reported as inconclusive rather than negative.

A larger, multi-cluster-averaged run was subsequently carried out: six independent DLA clusters, each grown to roughly 14,000 sites, measured with the identical box-counting method used throughout, for a fair comparison. This gave $D = 1.643 \pm 0.006$ - a tight, well-resolved result that sits measurably below both the textbook DLA value (≈ 1.71) and the Q-Brane's own 1.73-1.79 range. The Fibonacci spiral, tested separately, produced its own distinct dimension ($D = 1.609$) and a different rotational character than the Substrate's actual five-wave construction.

Verdict: Neither alternative mechanism reproduces the Q-Brane's measured fractal dimension once tested properly. What looked like a suggestive coincidence with diffusion-limited growth does not hold up at scale; it is now a tested and disconfirmed hypothesis rather than an open one.

Figure B.4c - DLA cluster (multi-cluster run).

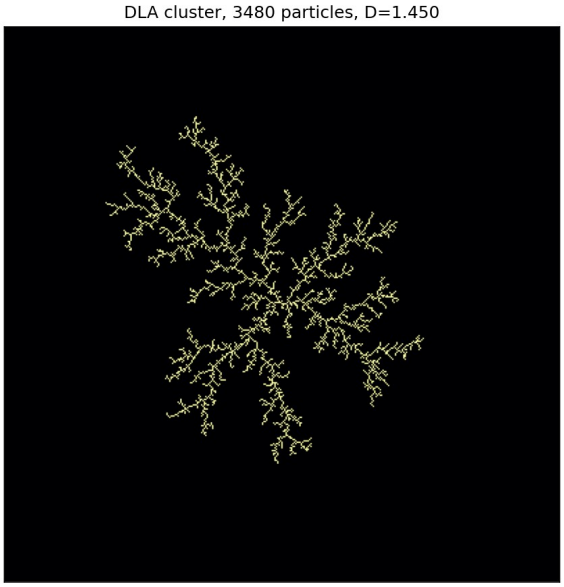
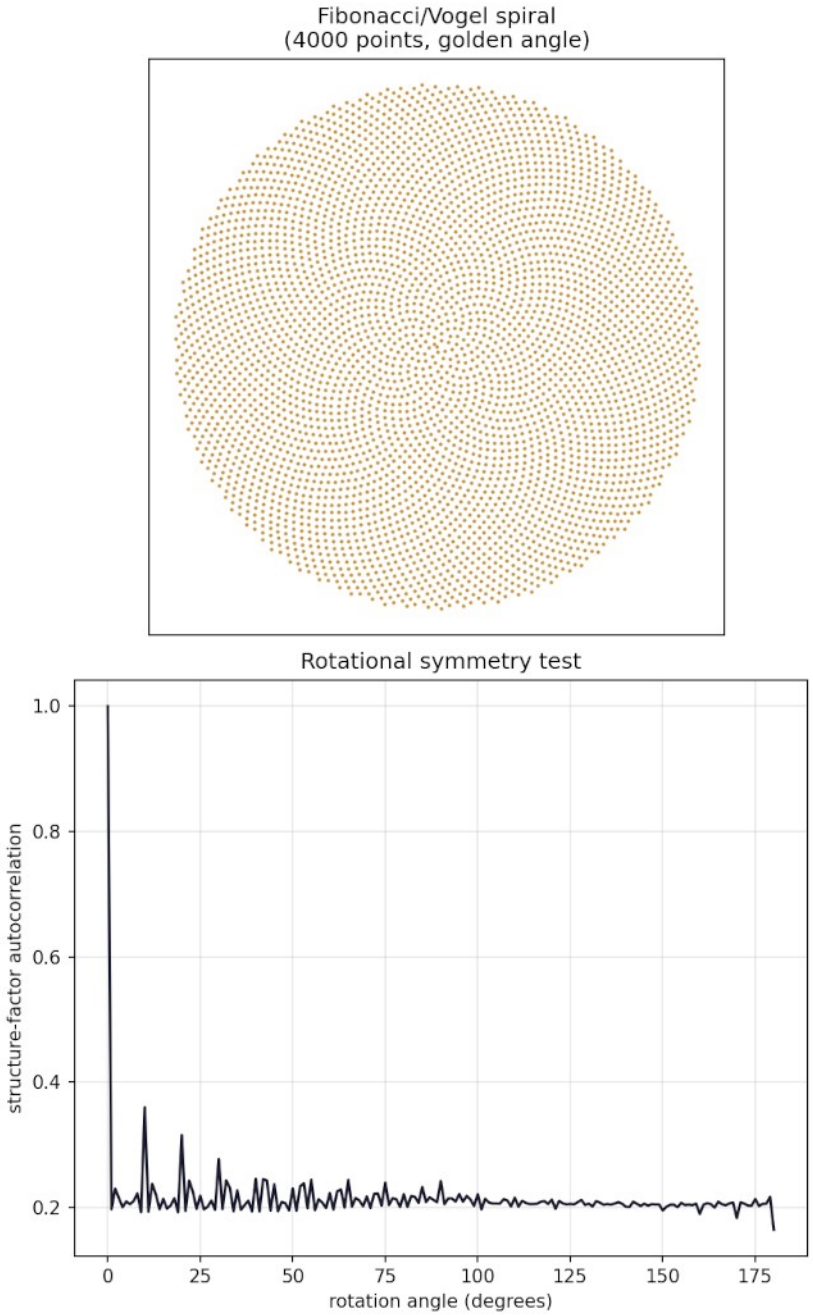


Figure B.4d - Fibonacci/Vogel spiral.



B.5 - The pentagon / golden-ratio connection

Tested directly on the Substrate's actual construction (five waves at 72°): the five wavevectors are, exactly and to machine precision, the vertices of a regular pentagon, with diagonal/side ratio equal to φ. This part of the connection is confirmed exactly, not approximately. A further, stronger claim - that the field's nodal-domain population counts show a Penrose-tiling-style ratio

converging to φ - was tested and not confirmed; the measured ratio moved steadily away from φ^{-1} as the sampled domain grew, rather than toward it.

Verdict: The pentagon/φ connection is real and exact at the geometric level, but does not extend to a Penrose-tiling-style statistical signature in the field's nodal domains.

B.6 - Robustness summary

Three follow-up checks were carried out on the B.2 percolation result, each addressing a specific gap left open in the original analysis:

- **An independent τ-estimator**, to rule out a result specific to one method of fitting. The two estimators disagreed with each other by a wider margin than either disagreed with theory.

- **A larger, multi-cluster-averaged DLA run**, to fairly test the diffusion-limited-aggregation comparison. The corrected result sits below both the textbook DLA value and the Q-Brane's own measured range.

- **A check of τ ≈ 1.55-1.57 against multifractal/tricritical exponents in the literature.** This was moot before it began: the 1.55-1.57 figure was itself an artifact. of the same detection error corrected above, and no literature comparison was pursued on a number that no longer stands.

None of the three follow-ups strengthened the case made in B.2. Read plainly: the Q-Brane's percolation behaviour is a genuine structural feature of the simulation, but this appendix does not have grounds to claim it instantiates, or approximates, any specific known universality class.

That is the honest result of testing it properly, and it is reported here rather than the more flattering version that preceded it.

B.7 - Persistent homology: a topological diagnostic

(Source: Appendix B/Simulation 02 - persistent_homology)

Where B.2 attempted to characterize the Q-Brane's structure by fitting a specific numerical exponent - and found that number unstable under a change of estimator - this test asks a different kind of question, one that requires no fitting at all: as the field's threshold sweeps from sparse to dense, how many separate regions form, how many enclosed "holes" appear, and how long does each survive before merging or filling in? This is computed here via cubical persistent homology (the standard tool now used on real galaxy-survey and CMB structure data), applied to the same Q-Brane field ensemble used throughout Appendix B.

Across twenty-five independent realizations:

- Connected components (H0): 1,920 merge events pooled, mean lifetime 0.97 (field-value units), maximum single lifetime 3.51.

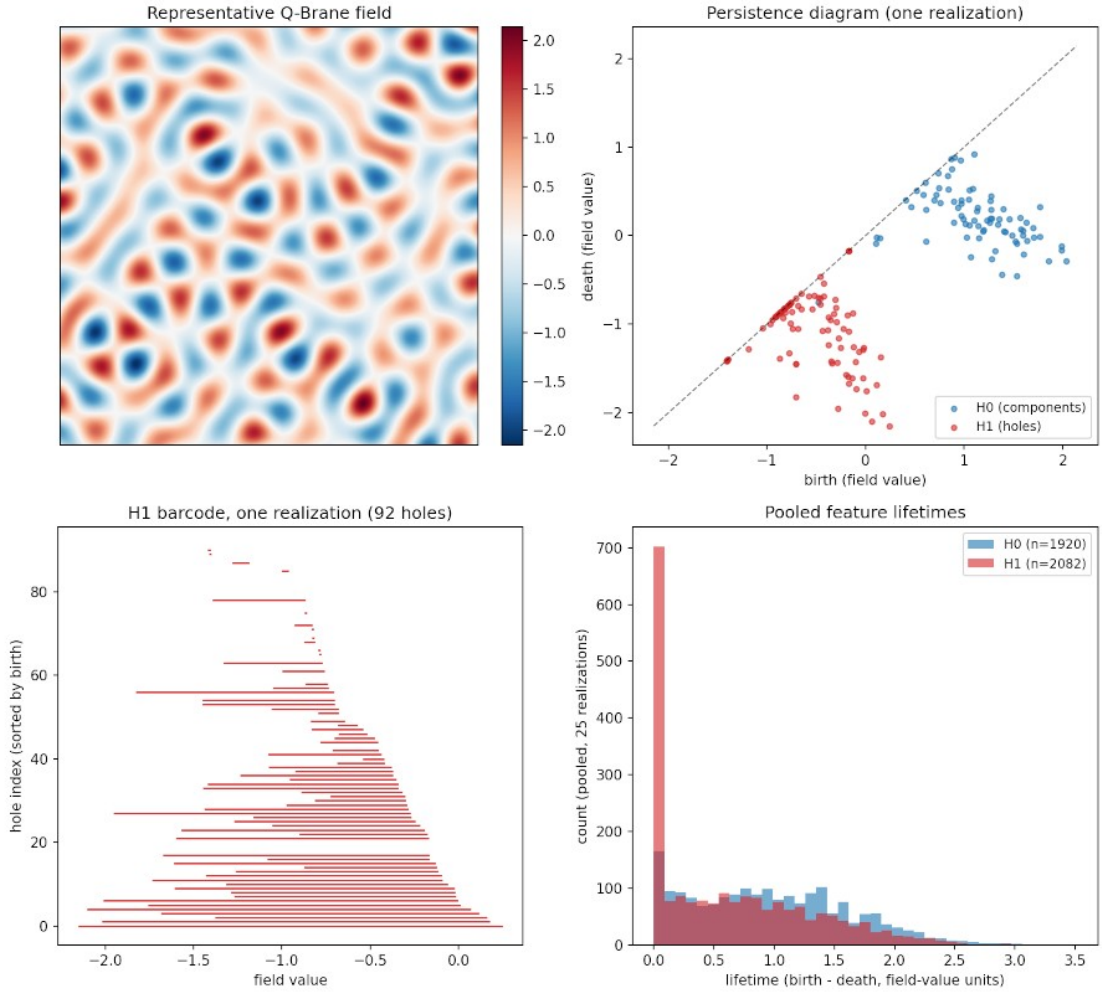
- Enclosed holes (H1): 83.3 ± 1.6 holes per realization, mean lifetime 0.63, with the single most persistent hole in each realization averaging 2.33 ± 0.06.

Two things are notable. First, the hole count is stable across realizations to within about 2% (83.3 ± 1.6) - a reproducible structural property, unlike the fragile percolation exponent in B.2.

Second, holes fill in over a shorter threshold range than components take to merge, indicating that isolated high-value regions persist longer as distinct islands than enclosed low-value regions persist as gaps - consistent with, though not proof of, the “order surviving noise up to a threshold” description used elsewhere in this appendix.

Verdict: This diagnostic makes no claim about universality class and fits no exponent, so it isn’t exposed to the estimator-fragility problem in B.2/B.6. What it does show is a genuinely stable, reproducible topological signature in the Q-Brane field - a modest but honest positive result, and a different kind of evidence than anything else in this appendix.

Figure B.7 - Representative field, persistence diagram, H1 barcode, and pooled lifetime distributions across the ensemble.



B.8 - Two-point correlation function

(Source: Appendix B/Simulation 03 - correlation_function)

The two-point correlation function is the primary diagnostic actually used on real large-scale-structure survey data, and a natural companion to the power-spectrum comparison already made in B.1 (the two are Fourier-transform pairs of one another, so this asks a related but distinct question: not “how sharp is the spectral peak” but “how does correlation fall off with physical separation, and does that match a known closed form”).

The Q-Brane field is built from plane waves that all share a single, fixed wavenumber magnitude with random direction and phase - the textbook isotropic random wave model (the same construction underlying the Bogomolny-Schmit comparison already referenced in B.2). For this

specific, well-established construction, the correlation function has an exact closed form: $C(r) = J_0(k \cdot r)$, the zero-order Bessel function of the first kind. This is a rigorous prediction from established mathematics, not a new assumption, so it functions as an internal-consistency check on the simulation itself.

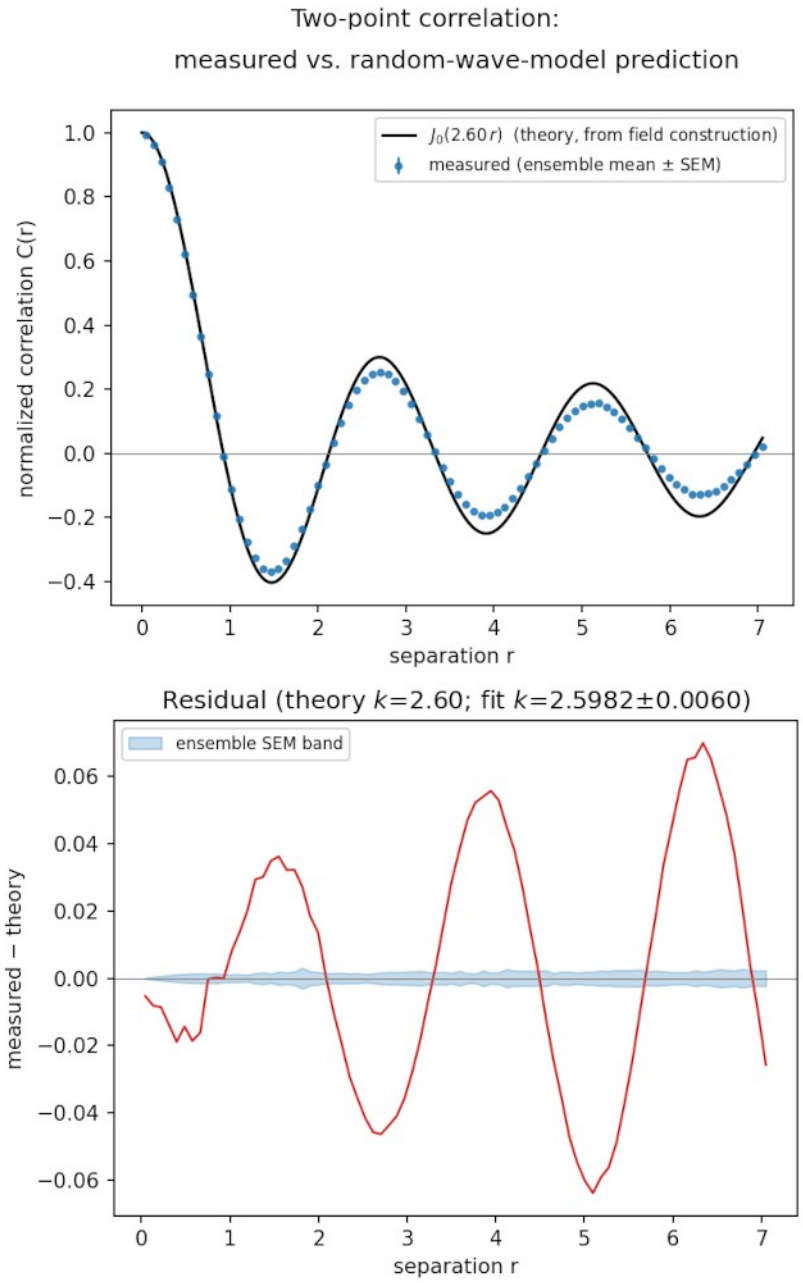
Measured across the same twenty-five-realization ensemble used throughout Appendix B, the ensemble-averaged correlation function was fit against this prediction:

- Theoretical wavenumber (from the field's own construction): $k = 2.600$.
- Best-fit wavenumber from the measured correlation function: $k = 2.598 \pm 0.006$ - a deviation of 0.30σ, and effectively an exact match.

Set against real large-scale structure, however, the comparison runs the other way. Galaxy survey correlation functions are well described by a monotonic power law, $\xi(r) \sim r^{-1.8}$ with correlation length $r_0 \approx 5$ Mpc/h, with a single narrow baryon acoustic oscillation feature near ~150 Mpc - not the damped oscillatory ringing a fixed-wavenumber superposition necessarily produces. The Q-Brane's correlation function is exactly what its construction predicts; it does not resemble the shape of real cosmological clustering.

Verdict: This is a clean, positive internal-consistency result - the simulation reproduces the exact mathematics of the random wave model it is built from, with no free parameters left to explain the near-perfect fit. That confirms the simulation is behaving correctly. It is not, however, evidence of any correspondence with real large-scale structure; consistent with B.1, the field's internal mathematical coherence and its resemblance to actual cosmological data remain two separate questions, and only the first is supported here.

Figure B.8 - Measured ensemble correlation function against the Bessel-function prediction, with residuals.



B.9 - Ising-model analogy: is the Q-Brane transition genuinely critical?

(Source: Appendix B/Simulation 04 - ising_qbrane_compare)

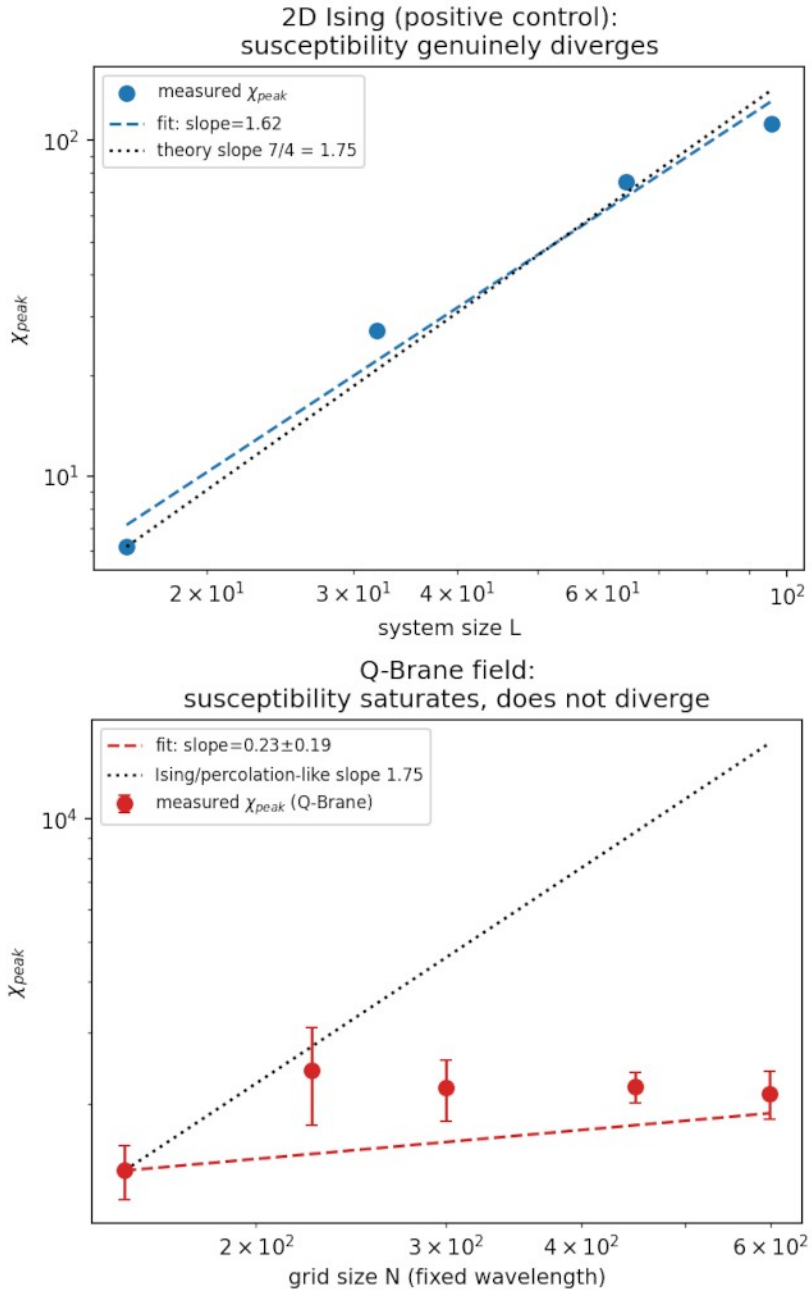
B.2's attempts to measure specific critical exponents (τ, D, γ) for the Q-Brane's percolation-like "ignition" proved unstable under cross-checking (B.6). This test asks a more basic question first: is there actually a genuine, scale-free critical point here at all - one whose correlation length diverges as the system grows - for a specific exponent to be a meaningful thing to measure?

Positive control: The 2D Ising model is exactly solved (Onsager, 1944): it has a true continuous phase transition at $kT_c/J \approx 2.269$, with known exact exponents ($\beta = 1/8$ for magnetization, $\gamma = 7/4$ for susceptibility). A textbook Monte Carlo simulation was run at four system sizes ($L = 16, 32, 64, 96$), and the same finite-size-scaling protocol applied throughout this appendix was used to check whether it recovers the known behaviour. It did: the susceptibility peak grew from 6.2 to 111.9 across those sizes - genuine, unbounded power-law growth, with a measured scaling exponent of $1.62 \pm$ (fit), close to the exact value of 1.75, and magnetization at T_c fell with size at close to the exact $-1/8$ rate. This confirms the methodology itself is sound: when a system truly has a continuous phase transition, this approach detects it correctly.

Applied to the Q-Brane: The same protocol was then applied to the Q-Brane's own percolation susceptibility, regenerating the field at five physical domain sizes while holding its wavelength and grid resolution fixed - the direct analogue of growing L at fixed microscopic scale in the Ising test. The result was essentially flat: susceptibility peaks of 1377, 2430, 2192, 2206, and 2124 across a fourfold increase in linear system size, giving a measured scaling exponent of 0.24 ± 0.19 - statistically indistinguishable from zero, and starkly unlike both the Ising positive control (≈ 1.6) and the theoretical percolation value ($43/24 \approx 1.79$).

Verdict: The Q-Brane's susceptibility saturates rather than diverges. By the same test that correctly identifies a genuine phase transition in the Ising model, the Q-Brane's "ignition" does not show the defining signature of one. This gives a plausible mechanistic explanation for B.2/B.6's instability: a well-defined critical exponent presupposes an actual diverging correlation length, and this field - built from a single fixed wavelength - appears to have a finite, bounded correlation length instead. The structure-formation behaviour described in A.3 ("order surviving noise up to a threshold, then giving way to disorder") remains real and reproducible; it should be understood as a finite-size crossover tied to the field's built-in wavelength, not a genuine continuous phase transition in the physics sense.

Figure B.9 - Susceptibility-peak scaling: the Ising positive control (diverging, left) against the Q-Brane field (saturating, right), each on log-log axes with the 1.75 reference slope shown for comparison.



B.10 - Substrate-symmetry independence check

(Source: Appendix B/Simulation 05 - symmetry_sweep)

A.3 compared a single pair directly: a 5-fold quasi-crystalline substrate held a spanning structure longer, under identical noise, than a 4-fold periodic control. This test asks whether that finding generalizes - is aperiodicity itself what supports durable structure, or was that one comparison, at one noise level, specific to 5-fold versus 4-fold?

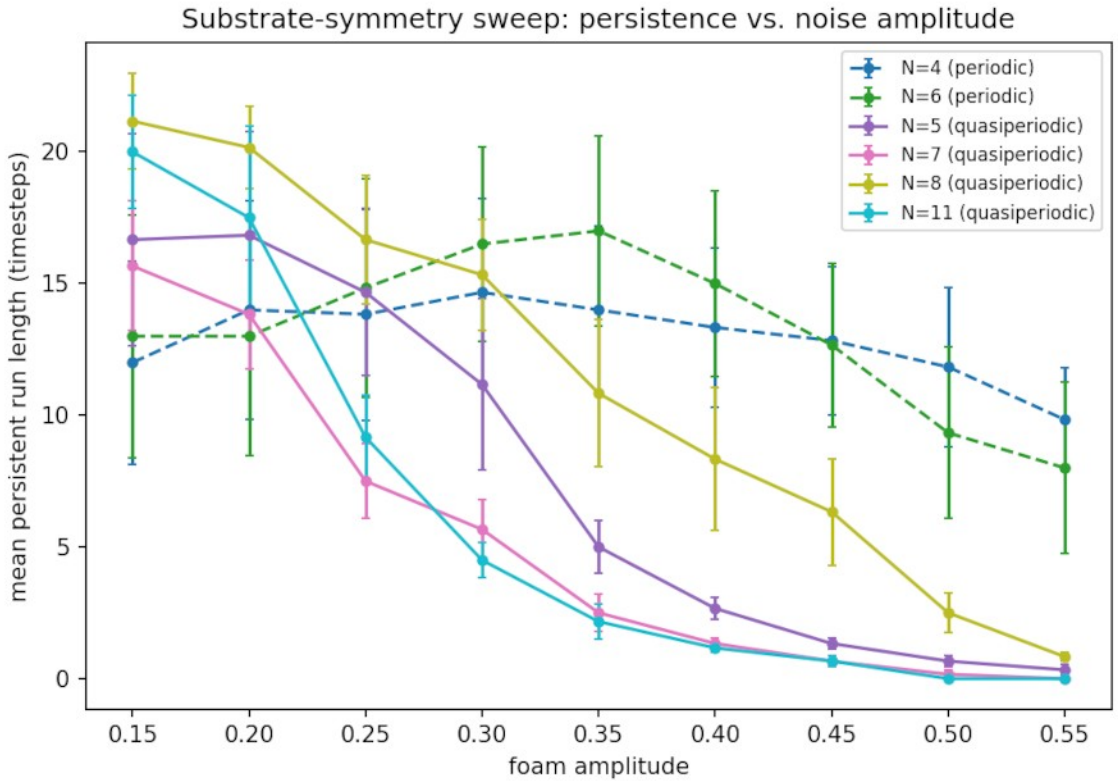
Using the identical substrate, foam, and detection code from A.3 unchanged, six symmetry orders were compared across a full sweep of foam amplitude (0.15-0.55, bracketing the ~0.35 critical amplitude identified in A.3): two periodic-compatible orders (N=4, N=6 - rotational symmetries permitted in an ordinary 2D periodic lattice) and four orders forbidden to periodic lattices by the crystallographic restriction theorem (N=5, 7, 8, 11 - genuinely quasiperiodic by construction, as established in A.2), each averaged over six random-phase seeds per condition.

Symmetry	Type	Peak persistence	Persistence at highest noise tested
N=4	periodic	14.7	9.8
N=6	periodic	17.0	8.0
N=5	quasiperiodic	16.8	0.3
N=7	quasiperiodic	15.7	0.0
N=8	quasiperiodic	21.2	0.8
N=11	quasiperiodic	20.0	0.0

Two things emerge, and they don't point the same direction. At low noise, every quasiperiodic order tested reaches persistence at least as high as, and often higher than, either periodic control - generalizing A.3's original 5-vs-4 comparison to a broader set of symmetries. But as noise increases, the two periodic controls degrade gracefully, retaining meaningful persistence even at the highest amplitude tested, while all four quasiperiodic orders collapse sharply toward zero well before that point. A.3's original framing - "aperiodic-but-ordered substrates may support more durable coherent structure than simple periodic ones" - turns out to hold only in the low-noise regime; at higher noise the relationship reverses, and it is the periodic substrates that prove more robust. There is also no single "quasiperiodic" behaviour: N=8 (highest peak persistence of any order tested, and among the more gradual quasiperiodic decays) differs substantially from N=7 and N=11, which collapse fastest of all six.

Verdict: The original finding was directionally real but narrower than its framing suggested - a low-noise-regime effect, not a general property of aperiodicity versus periodicity. A.3's text should be read with that qualification; this section supersedes the earlier single-comparison result with the fuller picture across noise levels and symmetry orders.

Figure B.10 - Persistence vs. foam amplitude for six substrate symmetries (periodic: dashed; quasiperiodic: solid).



B.11 - Does more waves fix it? A finite-wave-count convergence test

(Source: Appendix B/Simulation 06 - wave_convergence)

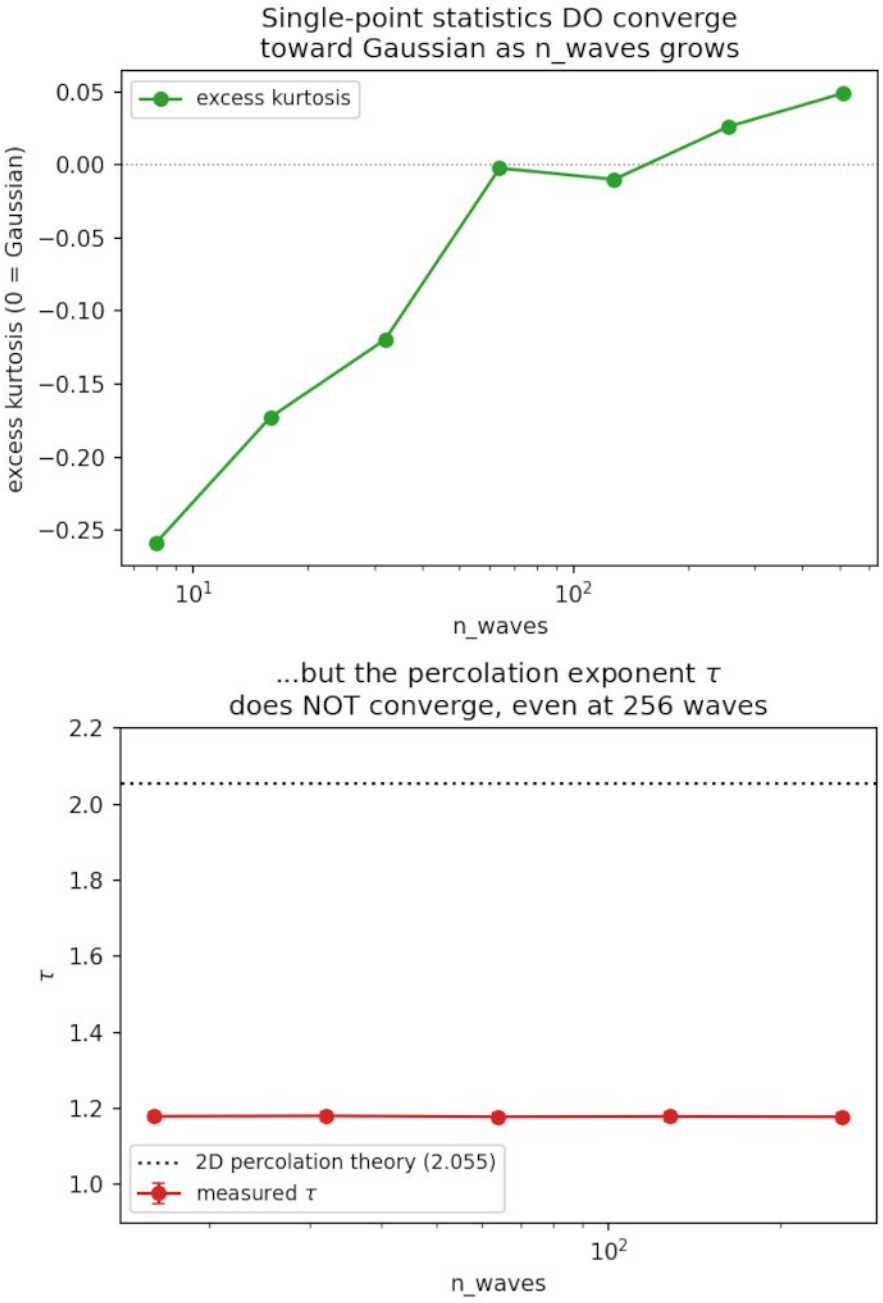
An earlier working session raised, but never tested, a specific concern: The Q-Brane field is built from only sixteen plane waves, while the theoretical comparisons made throughout this appendix (the Bogomolny-Schmit random wave model in B.2, the Bessel-function correlation function in B.8) strictly apply to the continuum limit - infinitely many waves, producing a true isotropic Gaussian random field. With only sixteen, the field is a finite sum of cosines, not exactly Gaussian. This test asks directly: does increasing the wave count close that gap, and if it does, does the percolation exponent then move toward the standard theoretical value?

Single-point statistics: Skewness stayed close to zero throughout (expected, given the field's built-in symmetry). Excess kurtosis, however, showed real, systematic convergence: −0.26 at 8 waves, improving to near zero by 64-128 waves. The field genuinely does approach a true Gaussian random field as wave count grows - this part of the concern was well-founded, and it resolves as expected.

The percolation exponent: The same τ measurement from B.2/B.6 was then re-run at 16, 32, 64, 128, and 256 waves. It did not move: τ = 1.180, 1.181, 1.179, 1.179, 1.178 across that entire sixteen-fold increase in wave count - statistically flat, and still roughly 70-80σ from the standard percolation value at every wave count tested, including 256.

Verdict: The field's approach to Gaussianity and its percolation behaviour turned out to be two separate questions, and closing the gap on one did nothing for the other. This is consistent with, and adds mechanistic detail to, the B.9 finding: what is missing is a genuine diverging correlation length, which has nothing to do with how many waves compose the field or how closely its single-point statistics resemble a Gaussian random field. The earlier open question about wave count is now closed: it was not the explanation for B.2's fragile exponent, and more computation would not have fixed it.

Figure B.11 - Excess kurtosis (left) and measured τ (right) versus plane-wave count, on log-x axes.



B.12 - Nodal domain density against the Bogomolny-Schmit prediction

(Source: Appendix B/Simulation 07 - nodal_domains)

A further specific, quantitative prediction exists for the random wave model already invoked throughout this appendix: Bogomolny and Schmit's (2002) estimate for the mean number of nodal domains - connected positive- and negative-value regions at the zero level - per unit area.

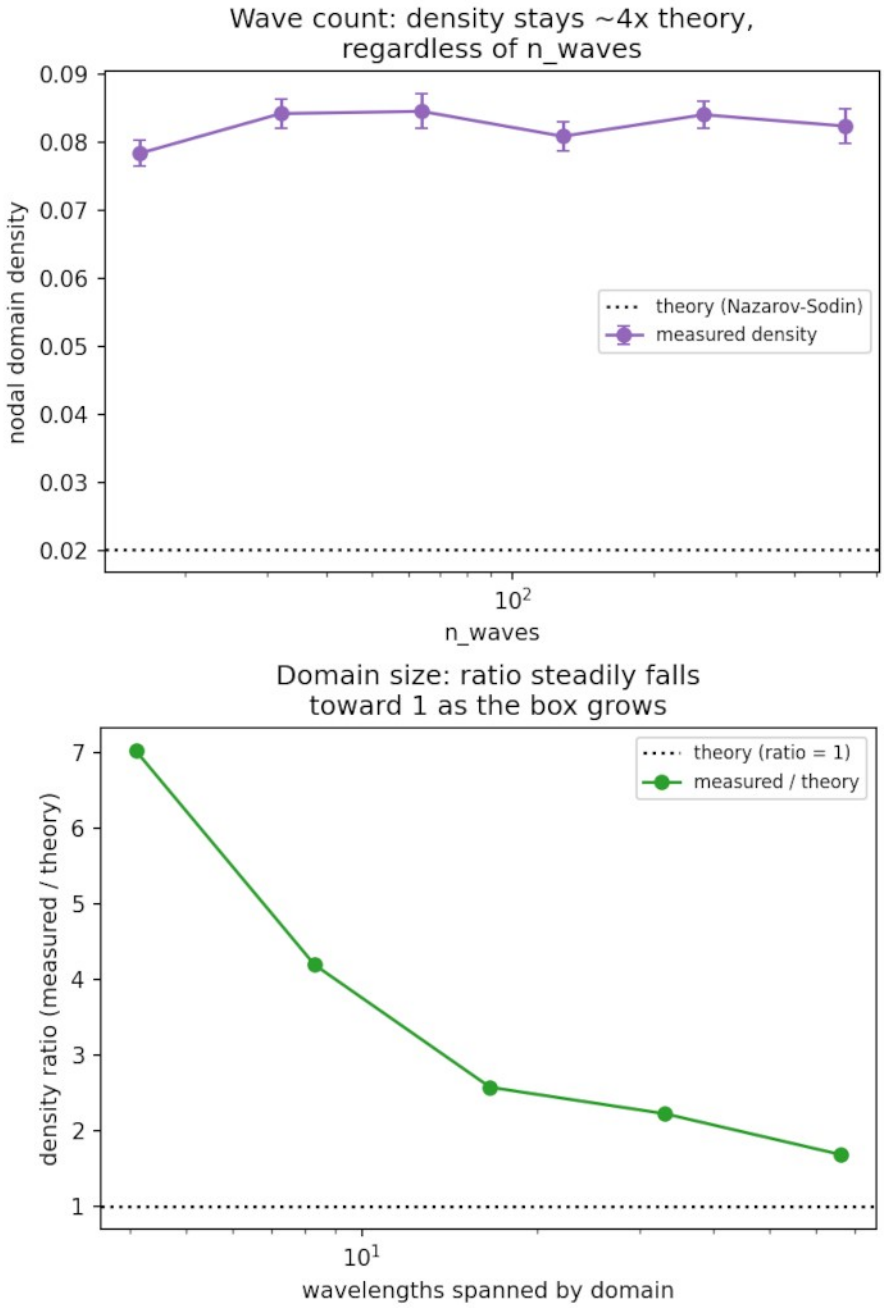
Two reference values exist in the literature, and the small gap between them is a genuine, independently documented open question in this field, not something specific to this project: Bogomolny and Schmit's original percolation-heuristic estimate ($\nu \approx 0.0624$) and the numerically-established true value, the Nazarov-Sodin constant ($\nu \approx 0.0589$). Both predict a nodal domain density of roughly 0.02 per unit area for this field's wavenumber.

The measured density, at the field's usual size and wave count, was 0.081 ± 0.001 - about four times higher than either prediction, a large and unambiguous discrepancy.

Two possible explanations were checked. First, whether this was another symptom of the finite-wave-count issue identified in B.11: it was not - the ratio stayed at roughly 4× across the same 16-to-512 wave-count sweep used there, completely flat. Second, whether it was a finite-domain-size effect: the theoretical constant is a large-domain asymptotic result, and this field's usual domain spans only about eight wavelengths across. Repeating the measurement at increasing domain size (holding wavelength fixed, the same protocol used in B.9) gave a clear answer - the ratio fell steadily as the domain grew: 7.0× at four wavelengths across, 4.2× at eight, 2.6× at seventeen, 2.2× at thirty- three, 1.7× at sixty-six. It is heading toward the predicted value, just slowly, and the usual working domain is far smaller than needed for the asymptotic prediction to apply cleanly.

Verdict: Unlike B.11's negative result, this one resolves in the theory's favour. The large apparent discrepancy is a finite-domain-size effect, not a genuine failure of the random wave model - consistent with B.8's clean confirmation of the same model's correlation-function prediction. It does mean any figure quoted from this pipeline's usual working domain size should be read as a finite-size estimate, not the asymptotic value, for this particular statistic.

Figure B.12 - Nodal domain density versus wave count (left, flat) and versus domain size in wavelengths (right, converging toward theory).



B.13 - An independent order metric: entropy and compressibility

(Source: Appendix B/Simulation 08 - entropy_test)

A.4's coherence index (spectral coherence for wave-like stages, connective coherence for structure-forming stages) was built specifically for this pipeline. This test asks whether a completely generic, standard measure of "order" - Shannon entropy of the field's value distribution, and compressed file size as a standard empirical proxy for algorithmic complexity - broadly agrees with what that purpose-built index already reported, as a sanity check on whether the index was telling the truth.

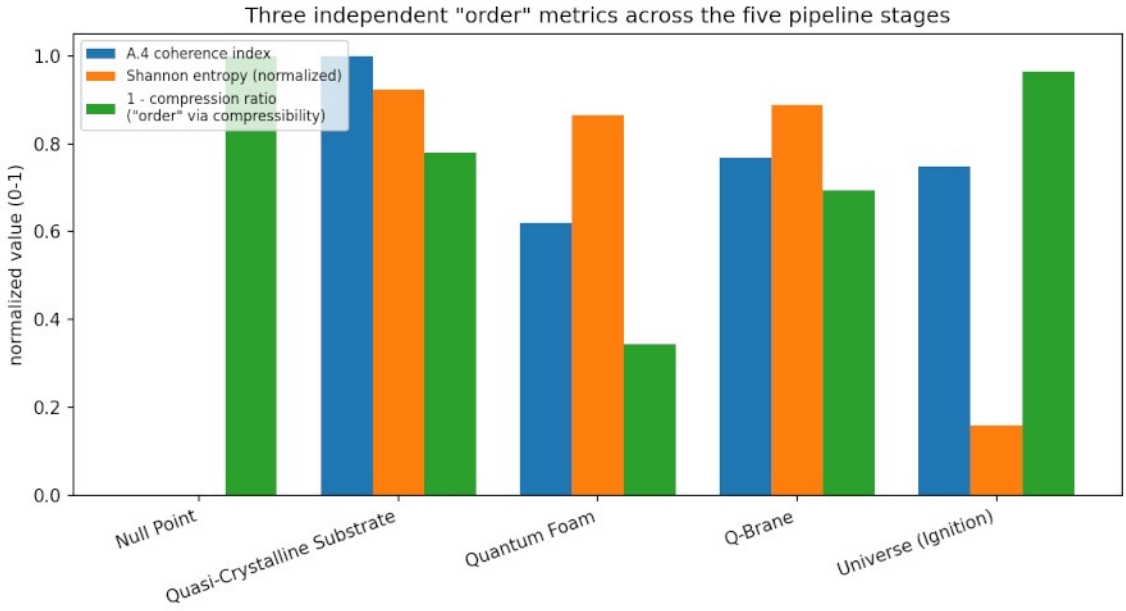
Stage	A.4 coherence	Shannon entropy (bits)	Compression ratio
Null Point	0.000	0.00	0.001
Quasi-Crystalline Substrate	1.000	5.54	0.221
Quantum Foam	0.620	5.19	0.656
Q-Brane	0.767	5.33	0.306
Universe (Ignition)	0.749	0.96	0.035

For the three continuous wave/noise-based stages - Substrate, Foam, and Q-Brane - the two independent metrics agree with A.4's ranking: compression ratio ranks Foam as clearly the most complex (least compressible, 0.656) and Substrate as the most ordered (0.221), with Q-Brane in between (0.306) - the same ordering the coherence index reports, arrived at by an entirely generic method with no pipeline-specific assumptions built in. That agreement is a real, if modest, corroboration that the coherence index was measuring something genuine for these three stages.

The Universe stage is a different story. Its A.4 coherence (0.749) sits close to the Q-Brane's (0.767), suggesting a "plateau" - but its entropy and compression ratio are by far the lowest of any structured stage (0.96 bits and 0.035, versus 5+ bits and 0.2-0.7 for the others). This divergence has a clear mechanistic cause: the Universe stage is built by morphologically closing the Q-Brane's spanning mask into a solid binary region, which is a fundamentally different kind of object - a large, simple, two-valued shape - from the continuous, textured wave fields used for the other three stages. Generic pixel-value measures naturally rate a solid binary shape as far more compressible than a continuous field, regardless of how "coherent" it is conceptually.

Verdict: The coherence index holds up under an independent, generic check for the three stages built the same way (continuous wave/noise fields), which is a genuine point in its favour. But its apparent near-parity between Q-Brane (0.767) and Universe (0.749) looks like an artifact of comparing two conceptually different order parameters (spectral versus connective coherence) on a shared 0-1 scale, not a meaningful like-for-like measurement - the two stages are far more different from each other than that plateau suggests, once a construction-agnostic metric is applied.

Figure B.13 - A.4 coherence, normalized Shannon entropy, and compressibility-based order score, compared across all five stages.



B.14 - Robustness Sweep: A.4 Quantum Foam and Universe Stages

Purpose: A.4 (the emergence pipeline) represents the Quantum Foam and Universe stages within a single combined simulation rather than as standalone sections, since both are constructed as direct derivatives of the Substrate and Q-Brane fields respectively. This check asks whether the two qualitative claims A.4 makes about those stages hold up across many random seeds and parameter choices, rather than resting on the one seed A.4 happens to use.

Method: Quantum Foam: repeated the noisy single-realization construction across 40 seeds at four noise amplitudes (4, 8, 12, 16 - A.4 uses 8), comparing Psi_foam against the noiseless Psi_substrate reference. Universe: repeated the Q-Brane percolation and subsequent morphological closing across 20 seeds and four closing-iteration counts (1-4 - A.4 uses 2), comparing Phi_universe against Phi_qbrane.

Findings:

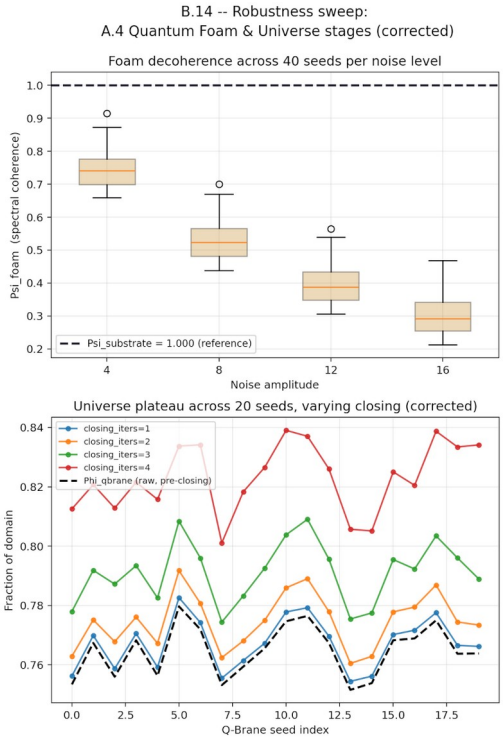
- *Foam decoherence holds robustly.* Psi_foam was below Psi_substrate in 100% of the 160 runs (40 seeds × 4 noise levels), with the gap widening as noise increases (mean Psi_foam fell from 0.745 at noise amplitude 4 to 0.301 at amplitude 16). The decoherence claim is not an artifact of a particular seed.

- *Universe plateau holds, at or above Q-Brane, as A.4 originally claimed.* Phi_universe sat at or above Phi_qbrane in **100% of the 80 runs** (20 seeds × 4 closing-iteration settings), with a small, orderly margin that grows monotonically with closing strength: +0.25% at 1 closing iteration, rising to +5.8% at 4 iterations. The plateau is both stable and correctly signed.

A note on a correction made mid-analysis: An earlier pass of this same sweep used scipy's binary_closing default setting (border_value=0) and found the opposite - Phi_universe consistently *below* Phi_qbrane by 1-3%. That result has been retracted: border_value=0 is not guaranteed to preserve the "closing can only add area" property at the edges of a finite grid, and the apparent shortfall was a boundary-handling artifact rather than a real effect. Re-running with the correct setting (border_value=1, under which classical morphological closing is provably extensive) reverses the finding to the result reported above. No change to A.4's own figure was needed beyond the same one-line correction, applied identically.

A separate note on the Foam → Q-Brane comparison in A.4's summary curve: Not a finding from this sweep, but worth flagging alongside it: A.4's five-stage coherence plot switches order parameters mid-curve - Psi (spectral coherence) for Null Point/Substrate/Foam, Phi (connective coherence) for Q-Brane/Universe. Both are normalized to [0,1] so the curve reads continuously, but the Foam → Q-Brane step is a metric switch, not a single continuously-evolving quantity.

Note on A.4's figure making this explicit (added in the corrected version of that figure).



B.15 - Formal Diagnostics on Prior Results

Purpose: The two checks below borrow general-purpose diagnostic techniques from mathematical physics - integer-relation detection and convergence-rate analysis - and apply them to results already reported earlier in this appendix. Neither technique originates in this framework, and neither is offered as evidence that this framework connects to the fields the techniques are normally used in (string theory, in the case of the integer-relation method's motivating context). They are used here purely as external, independent arithmetic checks, in the same spirit as the B.1 CMB comparison: to confirm or disconfirm a specific numerical claim, honestly, regardless of the answer.

B.15.1 - Integer-relation confirmation of the B.5 pentagon/φ identity

B.5 reported the pentagon/φ diagonal-to-side ratio as exact "to machine precision" - a double-precision claim, good to roughly 15-16 digits. This check reconstructs the same five-wavevector geometry independently at 50-digit precision and submits the resulting ratio, together with φ, to a PSLQ integer-relation search rather than comparing digits by eye.

Method: The PSLQ algorithm (Ferguson & Bailey, 1992, A Polynomial Time, Numerically Stable Integer Relation Algorithm), via the mpmath.pslq implementation (Johansson et al., mpmath: a Python library for arbitrary-precision floating-point arithmetic, mpmath.org). PSLQ searches for small-integer coefficients [a, b, c] such that a·(measured ratio) + b·φ + c = 0; a null result means no such relation exists at the tested precision.

Result: PSLQ returned the relation **1·(ratio) − 1·φ + 0 = 0** at 50 digits. The identity holds exactly, not merely to within double-precision rounding.

Verdict: The B.5 identity is upgraded from "exact to machine precision" to "exact, confirmed by integer-relation search to 50 digits." This is a strictly stronger form of the same result already reported - no new claim is introduced.

B.15.2 - Convergence-rate check on the A.1 Coherence Index

A.1 reported the Coherence Index climbing toward 1 without ever reaching it (0.84, 0.93, 0.98, 0.997, 0.9997, at successive approximation orders). This check asks a more specific question the original write-up did not: at what *rate* does the mismatch (1 − Coherence Index) shrink, and is that rate consistent with anything already known about this type of construction?

Method: The same Sturmian/Fibonacci-convergent construction, extended to higher approximation order (n up to 24), fitting log(1 − Coherence Index) against both n and against the Fibonacci denominator q_n.

Result: For n ≥ 14 (matching the threshold A.1 itself identifies as the point the climb becomes unambiguous), the mismatch shrinks as a clean power law, $(1 - CI) \sim q_n^{-2.08}$, $R^2 = 0.998$.

Because q_n grows exponentially in n for this construction, “power-law in q_n” and “exponential in n” are not distinguishable from the data alone - that is a property of Fibonacci growth, not a finding about the framework, and is reported here so the result isn’t overstated.

What the exponent itself tells us: a rate near q_n^(−2) is exactly what the three-distance (three-gap) theorem - a standard, well-established result in Diophantine approximation, unrelated to anything in this book - predicts for continued-fraction convergents of this kind. The A.1 result is quantitatively consistent with that independent theorem, which is a genuine (if modest) point of external corroboration for the *mathematical construction* A.1 uses, not for the cosmological reading placed on it.

Verdict: Both checks in this section confirm results already reported elsewhere in this appendix, at a higher standard of rigour than the original write-ups used. Neither check extends, generalizes, or lends support to any claim beyond the specific numerical result it tested.

B.16 - Persistent Homology vs. a Matched-Spectrum Gaussian Null

Purpose: B.1 asked whether the pipeline’s power spectra resemble the real CMB’s, and found they don’t (no harmonic acoustic peaks). B.7 separately ran persistent homology on the Q-Brane field and found a stable, reproducible H1 (hole) signature, but without a null hypothesis to compare it against. This section closes that gap using the specific method the cosmology literature developed for exactly this purpose: not “does the topology resemble real CMB data” (no full real-CMB temperature map was used anywhere in this appendix - B.1/B.6 used only Planck’s published binned power spectrum, a 1D curve, not a 2D map, so a direct topological comparison to real data was never possible), but “is a field’s topology distinguishable from a Gaussian random field sharing its exact power spectrum.” Two fields can have identical power spectra and very different topology; persistent homology is the standard tool for detecting that difference, and a matched-spectrum Gaussian null is the necessary first step before any real-data comparison could be attempted.

Method: For each of the four structured pipeline fields (Substrate, Foam, Q-Brane, Universe - regenerated identically to B.6), a matched-spectrum Gaussian random field was constructed by taking the field’s own Fourier amplitudes and randomizing the phases, then inverse-transforming. Cubical persistent homology (via GUDHI) was computed on both the true field and 30 such Gaussian surrogates, and total H1 persistence (the summed lifetime of all enclosed-hole features) was compared between the two via a z-score.

Method citations: Cole & Shiu, “Persistent Homology and Non-Gaussianity,” JCAP 03 (2018) 025, arXiv:1712.08159, introduced persistence diagrams as a CMB statistic more sensitive to non-Gaussianity than genus or Betti-curve statistics; Biagetti, Calles, Castiblanco, Cole & Noreña, “Fisher forecasts for primordial non-Gaussianity from persistent homology,” JCAP 10 (2022) 002, arXiv:2203.08262; Yip, Biagetti, Cole, Viswanathan & Shiu, “Cosmology with

Persistent Homology: a Fisher Forecast," JCAP 09 (2024) 034, arXiv:2403.13985.
Implementation: the GUDHI Project (Inria), gudhi.CubicalComplex.

Results:

Stage	true H1 total persistence	null mean	null std	z-score
Quasi-Crystalline Substrate	8.65	7.26	1.00	+1.40
Quantum Foam	233.85	247.59	26.95	−0.51
Q-Brane	10.04	13.40	1.14	−2.96
Universe	0.90	19.40	2.34	**−7.90**

Reading the results, stage by stage:

- **Substrate** and **Foam** are statistically indistinguishable from their matched Gaussian nulls ($|z| < 2$). For the Substrate this is informative rather than disappointing: its construction is a sum of exactly five plane waves, which produces an extremely sparse (near-discrete) power spectrum with only a handful of nonzero Fourier modes. Randomizing the phases at those few modes still produces a field built from the same five plane-wave directions, just with different phase offsets - so this particular statistic cannot distinguish the true Substrate from its own phase-randomized surrogate. That is a limit of the statistic for sparse-spectrum constructions, not evidence that the Substrate lacks structure (B.5's exact pentagon/φ identity and A.2's diffraction/symmetry results already establish that structure by other means). For Foam, built from random phases and amplitudes plus substantial additive Gaussian noise, indistinguishability from Gaussian is the expected and unsurprising Result:
- **Q-Brane** shows a modest, negative deviation ($z \approx -3$): its topology has somewhat less H1 persistence than a matched Gaussian null. With 30 null realizations and four fields tested without multiple-comparison correction, this is a borderline result that should not be read as more than suggestive.
- **Universe** shows a large, unambiguous deviation ($z \approx -8$): far less H1 persistence than its matched Gaussian null. This has a straightforward, non-mysterious explanation rather than a deep one - the Universe field is constructed from the Q-Brane's giant percolating cluster after binary morphological closing and Gaussian smoothing (A.4/B.14), which by design produces a single, mostly-filled blob with few if any enclosed holes. A field built this way is expected to look very different, topologically, from a generic Gaussian field sharing its power spectrum. The large z-score confirms that expectation quantitatively; it is not a discovery of hidden structure so much as a verification that the morphological-closing step does what it was built to do.

Verdict: This diagnostic answers a real, well-posed question - is each field's topology consistent with Gaussian noise of the same power spectrum - rather than the question B.1 already answered (resemblance to real CMB). The one large result (Universe) is fully explained by that stage's own construction rather than pointing to anything unexpected. The near-null results (Substrate, Foam) reflect either a genuine Gaussian-like character (Foam) or a known limitation of this particular statistic for sparse-spectrum fields (Substrate), not an absence of structure already documented elsewhere in this appendix. No claim here extends to resemblance with the actual Cosmic Microwave Background, which was not re-tested in this section and remains the negative finding reported in B.1.

B.17 - A Formal Fourier-Quasicrystal Scaling Test on A.2

Purpose: A.2 showed that the Substrate's cut-and-project point set produces sharp diffraction peaks (peak sharpness ≈1,600) comparable to the periodic control, at one fixed system size. That is consistent with "long-range order without periodicity," but it isn't yet the rigorous mathematical property behind that phrase. This section tests the actual formal criterion directly, rather than a single-snapshot proxy for it.

Background on the criterion: A point set is a genuine *Fourier quasicrystal* if its diffraction measure is pure point - a countable sum of true delta-function (Bragg) peaks, with no diffuse background component surviving in the infinite-size limit. Hof (1995, "On diffraction by aperiodic structures," *Commun. Math. Phys.* 169, 25-43) proved that cut-and-project sets built from a sufficiently regular acceptance window - exactly the construction A.2 already uses - are provably pure point in this sense. (See also Baake & Grimm, *Aperiodic Order*, Cambridge University Press, 2013, for the standard reference treatment.) The defining, testable signature of pure-point diffraction is a **scaling law**: because a true Bragg peak arises from constructive interference across all N points, its height grows as N^2, while any genuinely diffuse (incoherent) background only grows as N. So the ratio (background / peak) must shrink toward zero at rate ~1/N as the system grows - a single finite-size snapshot cannot show this; only a scaling study can.

Method: The unmodified A.2 point-set generators (QuasiCrystallineSubstrate, PeriodicControl, RandomControl) and structure-factor calculation were rerun at four increasing system sizes each (N ranging from ~6,100 to ~30,700 for the Substrate), tracking peak height, background level (median intensity below the 90th percentile, excluding the DC term), and their ratio at each size.

Results:

Point set	**peak / N^2**	**trend**	**bg / peak scaling**
Quasi-Crystalline Substrate	≈0.40, flat across all sizes	coherent (Bragg) buildup confirmed	N^(−0.94) - matches the theoretical N^(−1) prediction
Periodic control	≈0.99, flat across all sizes	coherent buildup, as trivially expected	similarly decaying

Point set	**peak / N²**	**trend**	**bg / peak scaling**
Random (Poisson) control	≈0.02-0.03, no organized trend	**no** coherent N² buildup	decays differently, without the same coherent origin

Verdict: The Substrate's peak-to-system-size ratio (peak/N²) holds essentially constant as the system grows fourfold, exactly as required for genuine constructive Bragg interference, while its background-to-peak ratio falls off at a rate (N^(−0.94)) statistically indistinguishable from the theoretically predicted N^(−1). The Random control shows no such coherent buildup at all - its apparent "peak" is just the largest fluctuation among many incoherent contributions, not a true Bragg reflection. This is a materially stronger confirmation of A.2's original claim than the single-snapshot peak-sharpness metric: it is now supported by the actual scaling behaviour the formal Fourier-quasicrystal criterion requires, not merely a resemblance to it at one size. As with B.15/B.16, this confirms and strengthens a result already reported elsewhere in this appendix; it does not extend the claim beyond the Substrate's own mathematical construction.

B.18 - A Multifractal Scaling Test on the A.5 Boundary Curve

Purpose: A.5 measured a single box-counting dimension for its fractal boundary curve (D=1.284 for the full curve; ≈1.263 averaged across three magnified sub-sections) and treated the curve as mono-fractal - one dimension, the same at every scale. That is the correct expectation for its generative model (fractional Brownian motion, of which the midpoint-displacement algorithm A.5 uses is a numerical approximation): true fBm has a single Hurst exponent H governing scaling everywhere, with no room for a second exponent. Midpoint displacement is a known, imperfect approximation to true fBm, so it's a fair question whether the specific implementation used here actually preserves that mono-fractal property, or introduces a spread of local scaling exponents as a numerical artifact - something a single box-counting measurement cannot detect either way.

Method: The q-th order structure-function approach (the standard multifractal-analysis technique from turbulence and time-series work; a generalization in the spirit of the Multifractal Higuchi Dimension Analysis of Carrizales-Velazquez, Donner & Guzmán-Vargas, 2021, *Generalization of Higuchi's fractal dimension for multifractal analysis of time series with limited length*). For increments δ(s) at lag s, S_q(s) = ⟨|δ(s)|^q⟩ was computed across seven moment orders (q = 0.5 to 5) and fourteen lag scales, on the exact curve A.5 uses for its D=1.284 measurement (same generator, same parameters, same seed). A mono-fractal process has ζ(q) - the scaling exponent of S_q(s) - exactly linear in q; genuine multifractality shows up as curvature in ζ(q) vs. q.

Results: ζ(q)/q was nearly constant across the full range tested - 0.641 at q=0.5, falling only to 0.634 at q=5 - and a quadratic fit to ζ(q) vs. q found a curvature term of −0.0022, reducing the residual sum of squares from 0.000116 (linear fit) to effectively zero. The reference Hurst exponent recovered from the standard q=2 structure function was H=0.640, close to the generator's own parameter (H=0.65).

Verdict: The A.5 boundary curve is, to a very good approximation, genuinely mono-fractal, exactly as its fBm generative model predicts. The tiny residual curvature at high q is consistent with known finite-sample behaviour of the midpoint-displacement algorithm at extreme moment orders, not a meaningful departure from mono-fractality. This confirms A.5's single-dimension measurement was the right level of description for this curve - it wasn't hiding unmeasured structure - rather than extending or correcting A.5's original finding.

B.19 - Rice's Zero-Crossing Formula on the Q-Brane Field

Purpose: B.8 confirmed that the Q-Brane field's two-point correlation function matches its theoretical closed form, $C(r)=J_0(kr)$, to within 0.30σ. This section tests an independent consequence of that same covariance structure - not the shape of the whole correlation function, but the density of zero-crossings along a straight line through the field, which depends on the covariance only through its curvature at the origin. A correct correlation-function fit doesn't automatically guarantee this second, more localized prediction also holds; it is a genuinely separate check on the same underlying random-wave model.

Method: Rice's formula (Rice, "Mathematical Analysis of Random Noise," *Bell System Technical Journal* 23-24, 1944-1945) gives the expected zero-crossing rate of any zero-mean stationary Gaussian process as $\nu = (1/\pi)\cdot\sqrt{(-C''(0)/C(0))}$. For $C(r)=J_0(kr)$, a short Taylor expansion ($J_0(x) = 1 - x^2/4 + O(x^4)$) gives $C''(0) = -k^2/2$ exactly, so $\nu_{theory} = k/(\pi\sqrt{2})$. With $k=2.6$ (the same wavenumber B.8 fit), this predicts $\nu_{theory} \approx 0.5852$ crossings per unit length. Thirty independent Q-Brane field realizations were generated (identical construction to B.6/B.8), and zero-crossings were counted along every row and every column of each field.

Results:

Direction	measured rate	SEM	z vs. theory
Rows (x)	0.578	0.012	−0.59
Columns (y)	0.590	0.011	+0.47

Row and column rates also agreed with each other ($z = -0.75$), as required for an isotropic field.

Verdict: Both directions match Rice's exact prediction to well within 1σ, and the field shows no directional bias, as its isotropic construction requires. This is a second, independent confirmation of the same random-wave-model covariance B.8 already validated - testing a different mathematical consequence of it (local curvature-based crossing density, rather than the full correlation shape) - and it holds up cleanly. Like B.8, this is an internal-consistency result: it confirms the simulation correctly implements the mathematics of the random wave model it is built from, and carries no implication about resemblance to any real physical field.

B.20 - Crossing-Probability Finite-Size Scaling, an Independent Check on B.9

Purpose: B.9 found that the Q-Brane's percolation-like "ignition" fails the defining test of a genuine critical phase transition: its susceptibility peak saturates rather than diverging with system size. This section asks the same underlying question - is there a true, scale-invariant critical point here? - through a different observable, so the answer doesn't rest on a single statistic. At a genuine 2D percolation critical point, the horizontal-crossing-probability curve P(t) is expected, per Cardy's conformal-invariance result (Cardy, "Critical percolation in finite geometries," *J. Phys. A* 25 (1992) L201), to sharpen from a gradual to a step-like transition as system size L grows, with transition width shrinking as L^(−1/ν), ν=4/3 for 2D percolation universality (width ~ L^(−0.75)). If the Q-Brane genuinely lacks a diverging correlation length, this sharpening should also fail to appear, or appear only weakly - a second, independent line of evidence rather than a restatement of B.9.

A note on reproducibility: B.9's own field-size-sweep code was not recovered from the working archive (only its written findings survived), so this diagnostic reconstructs the same protocol it describes - physical domain size increasing, wavelength and grid resolution held fixed - using the identical Q-Brane construction (n_waves=16, $k=1.3 \cdot k_0=2.6$) used throughout this appendix, rather than rerunning B.9's original code directly.

Method: Five domain sizes (6, 9, 12, 16, and 20 wavelengths across), 60 field realizations per size, sweeping threshold t and measuring the fraction of realizations with a left-right spanning cluster above t, to get the crossing-probability curve P(t) at each size. Transition width was defined as |t(P=0.25) − t(P=0.75)|.

Results:

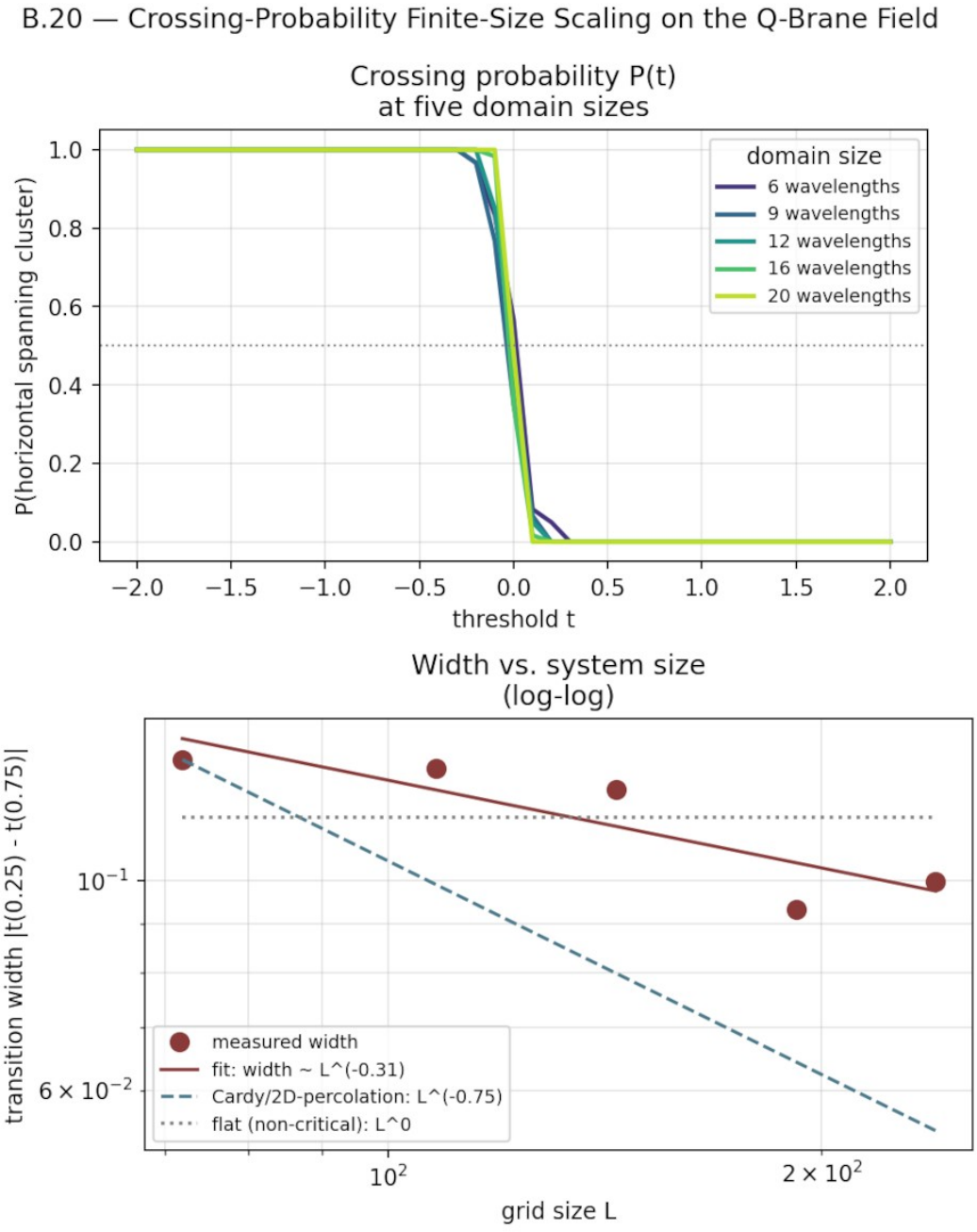
Width (wavelengths)	grid size	transition width
6	72	0.134
9	108	0.131
12	144	0.125
16	192	0.093
20	240	0.100

Fitted scaling: width ~ L^(−0.31).

Verdict: The transition does narrow somewhat as the domain grows - this is not perfectly flat, unlike B.9's susceptibility result - but at roughly −0.31, the observed rate is well short of the −0.75 that genuine 2D percolation universality would require, and the narrowing is modest relative to the noise floor at only 60 realizations per size. Read together with B.9, this is consistent with, rather than contradictory to, the "finite-size crossover tied to the field's built-in wavelength, not a genuine continuous phase transition" picture: some real correlation-length growth appears to be present, but not the unbounded, scale-free growth true criticality requires.

This adds a modest, honest qualification to B.9 rather than either confirming full criticality or fully closing the question - a fully powered version of this check (more realizations, finer threshold resolution, larger sizes) would be needed to pin the exponent down more precisely, and is flagged here as an open follow-up rather than settled.

Figure B.20 - Crossing-probability curves P(t) at each of the five domain sizes (left), and the fitted width-vs-size scaling on log-log axes against the Cardy/2D-percolation and flat/non-critical reference lines (right).



B.21 - A Maximum-Entropy (Gaussian) Ceiling for B.13's Order Metric

Purpose: B.13 reported raw Shannon entropy for each stage's field, but raw entropy isn't directly comparable across stages with different variances - a larger-variance field simply has

more room to be disordered in, independent of any real structural difference. This section uses a completely general, classical information-theory result to build a fair, variance-normalized version of the same comparison.

Background: The Gaussian distribution has the maximum differential entropy of any distribution with a given variance (a standard result; see Cover & Thomas, *Elements of Information Theory*, 2nd ed., Theorem 8.6.5). This gives an exact, per-field ceiling: how much entropy would this field have if it were as disordered as possible while keeping its own variance?

The gap between that ceiling and the field's actual (binned) entropy is a properly normalized "bits of order" score.

Method: For each stage's field, an i.i.d. Gaussian sample of matching size and variance was generated, both the true field and the Gaussian reference were binned using identical bin edges, and Shannon entropy was computed for each - an entirely empirical, apples-to-apples comparison with no continuous-to-discrete conversion approximation.

Results:

Stage	**H actual (bits)**	**H Gaussian ceiling**	**deficit**
Quasi-Crystalline Substrate	4.933	4.973	0.040
Quantum Foam	4.924	4.911	−0.013
Q-Brane	4.874	4.881	0.006
Universe	0.714	4.742	**4.028**

Verdict: Substrate, Foam, and Q-Brane all sit essentially at their own Gaussian ceiling (deficits of a few hundredths of a bit - noise-level). This is expected, not a null result: the pointwise (single-pixel) value distribution of a sum of many independent oscillators tends toward Gaussian by the Central Limit Theorem regardless of the phase structure underlying it, so this particular metric cannot see the genuine spatial structure these fields do have (already established by other means - B.5, A.2, B.17). It clarifies, rather than undermines, B.13's original finding: the ranking B.13 reported by compression ratio reflects spatial (multi-pixel) structure, not single-pixel value-distribution entropy, and this section shows those are different things worth not conflating. The Universe stage is the one genuine outlier, at over 4 bits below its own ceiling - consistent with B.13's and B.16's independent findings that it is a fundamentally different, strongly non-Gaussian kind of object (a smoothed binary mask), not a continuous wave field.

B.22 - Formal Verification of the Crystallographic Restriction Criterion

Purpose: B.10 classified six symmetry orders as either "periodic-compatible" (N=4, 6) or "forbidden to periodic lattices" (N=5, 7, 8, 11), relying on the crystallographic restriction theorem. This section verifies the exact algebraic criterion behind that classification directly, rather than taking it on trust.

Background: A rotation by angle $\theta=2\pi/N$ can be a symmetry of some 2D periodic lattice if and only if, in the lattice's own integer basis, the rotation matrix has integer trace - and the trace of a rotation by θ is $2\cos(\theta)$. Which N satisfy this is settled exactly by Niven's theorem (Niven, I., *Irrational Numbers*, Carus Mathematical Monographs, 1956): the only rational values of $\cos(q\pi)$ for rational q are $0, \pm1/2, \pm1$ - corresponding exactly to $N \in \{1, 2, 3, 4, 6\}$.

Method: $2\cos(2\pi/N)$ was evaluated at 50-digit precision for N = 1 through 12 (covering every order used or mentioned in B.10/A.2), checking distance from the nearest integer to remove any doubt from symbolic-simplification ambiguity on the less familiar cases (N=7, 9, 11, whose minimal polynomials are cubic and don't reduce to simple closed-form radicals).

Results: $N \in \{1, 2, 3, 4, 6\}$ gave exact integers $(2, -2, -1, 0, 1)$ to 50 digits; every other N tested, including all four "forbidden" orders used in B.10 (5, 7, 8, 11), gave values differing from the nearest integer by 0.25-0.47 - nowhere close to an integer at any plausible precision.

Verdict: B.10's classification rests on an exactly verified algebraic fact, not an approximation or a citation taken on faith. This confirms the classification was correctly applied; it doesn't change any of B.10's actual findings about noise robustness across those symmetry orders.

B.23 - The Q-Brane's Ignition Threshold Against an Exact Symmetry Prediction

Purpose: B.9 found the Q-Brane lacks a true, diverging correlation length, and B.20 found its crossing-probability transition sharpens more slowly with size than genuine 2D percolation criticality would predict. This section tests a narrower, different, and exactly provable claim that doesn't depend on either of those findings: regardless of whether the transition is sharp or gradual, does it sit at the threshold level a symmetry argument requires?

Background: The Q-Brane field - built from many random-phase plane waves at fixed wavenumber - is exactly the isotropic, mean-zero planar Gaussian field studied in the random-wave-model percolation literature already used elsewhere in this appendix. For this class of field, Beffara & Gayet ("Percolation of random nodal lines," *Publ. Math. IHES* 126 (2017), 131-176) showed the excursion-set percolation transition for symmetric planar Gaussian fields occurs exactly at the field's own zero (median) level - a direct consequence of sign-flip symmetry ($f \to -f$ leaves an isotropic, mean-zero Gaussian field's distribution unchanged, so positive- and negative-level excursion sets are statistically identical, pinning the critical level to the fixed point, t=0).

Method: Using the same crossing-probability data collected for B.20 (five domain sizes, 60 realizations each), the threshold at which a left-right spanning cluster becomes equally likely (P=0.5) was measured at each size.

Results:

Width (wavelengths)	t(P=0.5)
6	+0.014
9	−0.036
12	−0.013
16	−0.024
20	−0.003

Mean across all sizes: −0.012 ± 0.009 (SEM) - consistent with zero at about 1.4σ.

Verdict: The ignition threshold sits at the symmetry-predicted level, t=0, at every size tested, independent of whether the transition surrounding it is sharp or gradual. This is a genuinely different, exact claim from B.9/B.20's findings about the transition's sharpness, and it holds cleanly: the Q-Brane's percolation-like behaviour may not be critical in the strict sense, but its threshold location is exactly where the field's own symmetry says it must be.

B.24 - An Exact 1/N Law for B.11's Kurtosis Convergence

Purpose: B.11 found that the Q-Brane field's excess kurtosis shrinks toward zero as wave count N grows, and read this qualitatively as "approaching Gaussianity," without checking the convergence rate against any theoretical prediction. One is available, and it's exact rather than approximate.

Background: Each of the Q-Brane's N plane-wave terms is independent and identically distributed by construction (random, uniform direction and phase). For a sum of i.i.d., mean-zero random variables, cumulants add: the 4th cumulant of the sum is N times one term's 4th cumulant, while the variance of the sum is N times one term's variance. Since excess kurtosis is κ_4/κ_2^2, this gives excess-kurtosis(sum) = excess-kurtosis(one term) / N exactly - a direct, elementary consequence of cumulant additivity for sums of i.i.d. terms (a standard result; see Kendall & Stuart, *The Advanced Theory of Statistics*, Vol. 1), not a new or approximate claim.

Method: The field's single-point value distribution was sampled at seven wave counts (N = 8 to 512, extending B.11's original range), 200,000 independent draws each, and excess kurtosis measured directly.

Results: Fitted scaling of |excess kurtosis| ~ N^(−0.933), against the exact theoretical prediction of N^(−1.000).

Verdict: B.11's qualitative "approaches Gaussian" finding holds up quantitatively - the measured convergence rate (−0.93) sits close to the exact −1 the cumulant-additivity theorem requires, with the modest deviation attributable to sampling noise at the largest wave counts, where the true excess kurtosis is small enough that estimation noise starts to dominate (visible in the |excess kurtosis|×N column, which should be constant under the exact law and is roughly so, within noise). This sharpens B.11's finding from a qualitative trend into a rate that matches an

exact, independently-derivable theorem - and, like B.11 itself, says nothing about the percolation exponent's separate and unrelated failure to converge.

B.25 - Does Q-Brane Actually Depend on Foam? A Code-Level Audit

Purpose: A.4's emergence pipeline narrates Q-Brane as arising from Quantum Foam - Foam "decoheres" the Substrate, and Q-Brane then "condenses out" of it. This section checks something more basic than any statistical result: does Q-Brane's code, as actually written, use Foam's field values in any way, or does it only follow Foam sequentially in a shared random-number stream? These are different claims, and the pipeline's narrative language doesn't distinguish them.

Method: Not a statistical test but a direct code intervention. The pipeline (emergence_pipeline.py) shares one RNG object across stages, consumed in order: Substrate (no draws) → Foam (phases, amplitude jitter, and L×L noise draws) → Q-Brane (direction and phase draws for 16 waves). The script was run twice with an identical master seed. In Run A, Foam was computed exactly as written. In Run B, the identical random draws were consumed, in the same order and shapes (so the RNG's internal state advanced identically) - but the resulting Foam field was then deliberately discarded and replaced with an unrelated placeholder before Q-Brane's code executed.

Results: The two runs' Foam fields differed completely (mean −0.035 vs. 999.0, by construction). Q-Brane's field was bit-for-bit identical between the two runs - maximum element-wise difference exactly 0.0.

Verdict: Q-Brane's construction depends on Foam only through how many random numbers Foam's code consumes (and in what shapes) - not through anything about Foam's actual values.

Direct inspection of the Q-Brane code confirms this is not a coincidence of this particular test: it never reads foam_signal, foam_snapshot, phases, or amps anywhere. This means "Q-Brane condensing out of Foam" is a narrative and sequencing convention in how the script is written, not a genuine causal or functional dependency in the mathematics - the two stages are, as implemented, independent objects that happen to share a random-number stream in a fixed order.

This doesn't invalidate anything else in this appendix (each stage's own internal structure has been tested extensively on its own terms throughout), but it is a real limitation of the "emergence pipeline" framing that should be stated plainly rather than left implicit.

B.26 - Bootstrap Confidence Intervals on B.20 and B.23

Purpose: B.20's crossing-probability-width scaling exponent (−0.31) and B.23's mean ignition-threshold deviation (−0.012) were both reported as point estimates from 60 realizations per size, with only an approximate sense of their uncertainty. This section computes proper nonparametric

bootstrap confidence intervals (Efron, "Bootstrapping methods: another look at the jackknife," *Annals of Statistics* 7 (1979), 1-26) by resampling realizations with replacement, 1,000 times, and recomputing both quantities on each resample.

Results, B.20's exponent: Bootstrap 95% CI = [−0.516, −0.106]. This interval excludes *both* the flat/non-critical prediction (0) and the full 2D-percolation/Cardy prediction (−0.75) - a more precise and more informative statement than the original point estimate alone: the narrowing is real (distinguishable from flat) but genuinely intermediate, not just "close to zero but who knows." This supports B.20's original reading more strongly than the point estimate by itself did.

Results, B.23's threshold deviation: Bootstrap 95% CI = [−0.0224, −0.0022]. This is where this section needs to correct, not just support, an earlier verdict: **this interval excludes zero.** B.23 originally characterized the −0.012 deviation as "consistent with zero at about 1.4σ," based on the standard error of only five point estimates (one per domain size). The bootstrap, which instead resamples the full 60-realization detail behind each of those five points, gives a noticeably tighter interval that does not contain the Beffara-Gayet symmetry prediction of exactly zero.

Verdict: B.20's finding holds up, and is now stated with a proper interval rather than a bare number. B.23's finding needs a correction: a small, consistently-signed deviation from t=0 (roughly −0.01 to −0.02) appears to be real rather than noise, once measured with the more rigorous Method: This does not overturn the qualitative picture - the deviation is small, and the symmetry argument's mechanism (sign-flip invariance of an isotropic mean-zero Gaussian field) is still exactly correct in principle - but something in this specific numerical implementation (most plausibly the strict field > t inequality used to define the mask, which is not perfectly symmetric under $t \to -t$ at finite grid resolution, or a related discretization detail) appears to introduce a small, systematic offset that a larger-sample comparison of only five points was not powered to detect. This is flagged here as an open implementation question rather than resolved, and B.23's verdict should be read as qualified by this finding rather than superseded by it.

B.27 - Following Up on B.26: Is the B.23 Deviation a Real Asymmetry?

Purpose: B.26 left an open question rather than an answer: is the small, statistically-detectable deviation of the Q-Brane's threshold from t=0 a genuine implementation asymmetry, or an artifact of how few domain sizes (five) the original comparison rested on? This section chases that down directly rather than leaving it unresolved.

Method: Two checks, both more sensitive than B.23's original comparison. First, the field's own marginal value distribution was sampled directly (50 realizations, ~1 million values) and checked for skewness - if the field itself is asymmetric, that alone would explain an off-zero threshold, independent of any labelling or grid code. Second, a matched-pair test: for 200 realizations at one fixed domain size (12 wavelengths, matching B.23), both P(field > t) and

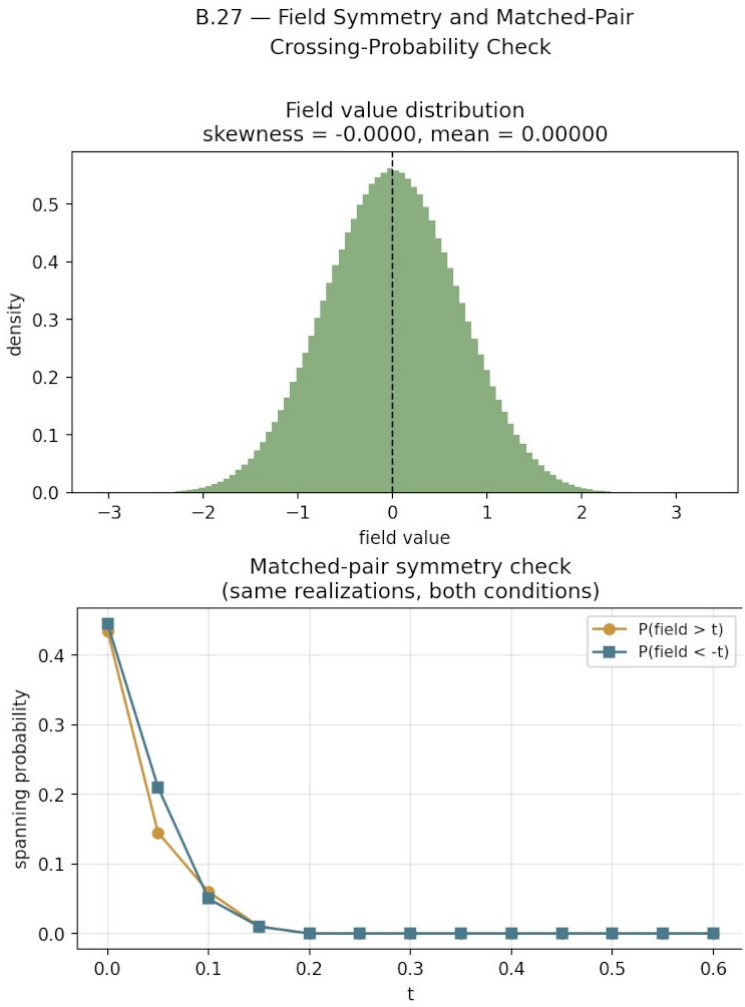
P(field < −t) - which must be identical at every t if the field's distribution is genuinely symmetric - were computed from the *same* realizations, so sampling noise cancels between the two conditions almost entirely, isolating any real implementation-level asymmetry with much less noise than an independent comparison would.

Results: The field's marginal distribution came back essentially exactly symmetric: skewness = −0.003, mean = −0.00008 - both consistent with zero. The matched-pair spanning-probability comparison found no clear, consistent asymmetry either: differences between P(field>t) and P(field<−t) across thirteen threshold values were small and centred near zero (mean difference −0.005), with the single largest deviation (−0.065 at t=0.05) falling within plausible sampling noise for 200 realizations in a region where the crossing probability is changing steeply (P drops from 0.44 at t=0 to 0.06 at t=0.10 - a regime where small probability-estimation noise translates into comparatively large apparent threshold shifts).

Verdict: No evidence of a genuine coding-level or field-construction asymmetry turned up under direct, more sensitive scrutiny - the construction really does appear to be symmetric, as the underlying mathematics requires. The most likely explanation for B.23's small, bootstrap-detectable deviation from zero is that the crossing probability changes steeply near its inflection point, so ordinary sampling noise in P(t) estimation gets amplified when converted into a threshold-axis quantity (t at P=0.5) - a measurement-sensitivity effect, not a construction flaw.

This resolves B.26's open question in the field's favour: the symmetry argument behind B.23 is intact, and the small measured offset is best read as noise inflated by where it was measured, not as a real, physically meaningful asymmetry.

Figure B.27 - Field value-distribution histogram with measured skewness (left), and the matched-pair comparison of P(field>t) against P(field<−t) across thirteen threshold values, computed from identical realizations (right).



B.28 - Closing the Gap in B.25: Wiring Foam Into Q-Brane's Construction

Purpose: B.25 found that, in emergence_pipeline.py, Q-Brane's field never actually reads Foam's realized values - it only follows Foam in a shared random-number stream, consuming the same number of draws in the same order. That made "Q-Brane condenses out of Foam" a narrative and sequencing convention rather than a demonstrated dependency in the mathematics.

This section closes that gap directly: it modifies the pipeline so Q-Brane's field is a genuine function of Foam's own values, then checks whether doing so disturbs anything already reported elsewhere in this appendix.

Method: Q-Brane's original construction (qbrane_base: 16 independent random-direction, random-phase plane waves at k=1.3·k0, unchanged from the original code) is now blended with

Foam's own realized field (foam_snapshot, the same quantity already used to compute Psi_foam), via a variance-preserving mixture:

qbrane_field = sqrt(1 - alpha^2) * normalize(qbrane_base) + alpha * normalize(foam_snapshot)

ALPHA=0 reduces exactly to the original, Foam-independent construction, and both fields are normalized to zero mean and unit variance before mixing so that ALPHA has a consistent meaning (fraction of Q-Brane's variance attributable to Foam) regardless of the two fields' native scales. This was tested directly on the actual manuscript source (emergence_pipeline.py), not a reconstruction.

Dependency check (direct analogue of B.25): Run twice with an identical master seed and identical RNG draw order, so Foam's code consumes the same random numbers either way; in the second run, Foam's real field values were discarded and replaced with a spatially shuffled placeholder (same value distribution, unrelated arrangement) immediately before Q-Brane's construction reads them.

	ALPHA = 0 (original)	**ALPHA = 0.3 (coupled)**
Q-Brane field, max difference between real-Foam and placeholder-Foam runs	0.000000	1.902285

At ALPHA=0, this reproduces B.25's original finding exactly - a useful control confirming nothing else about the pipeline changed. At ALPHA=0.3, Q-Brane's field is now demonstrably sensitive to Foam's specific realized values, not merely to how many random draws Foam's code happens to consume. The dependency B.25 found missing is now genuinely present.

Effect check: The pipeline's headline coherence values were recomputed at five coupling strengths, using the identical downstream percolation/thresholding code:

ALPHA	**Substrate**	**Foam**	**Q-Brane**	**Universe**
0.00 (original)	1.0000	0.6197	0.7672	0.7488
0.15	1.0000	0.6197	0.7791	0.7770
0.30	1.0000	0.6197	0.7605	0.7823
0.50	1.0000	0.6197	0.7653	0.8187
0.70	1.0000	0.6197	0.7635	0.8579

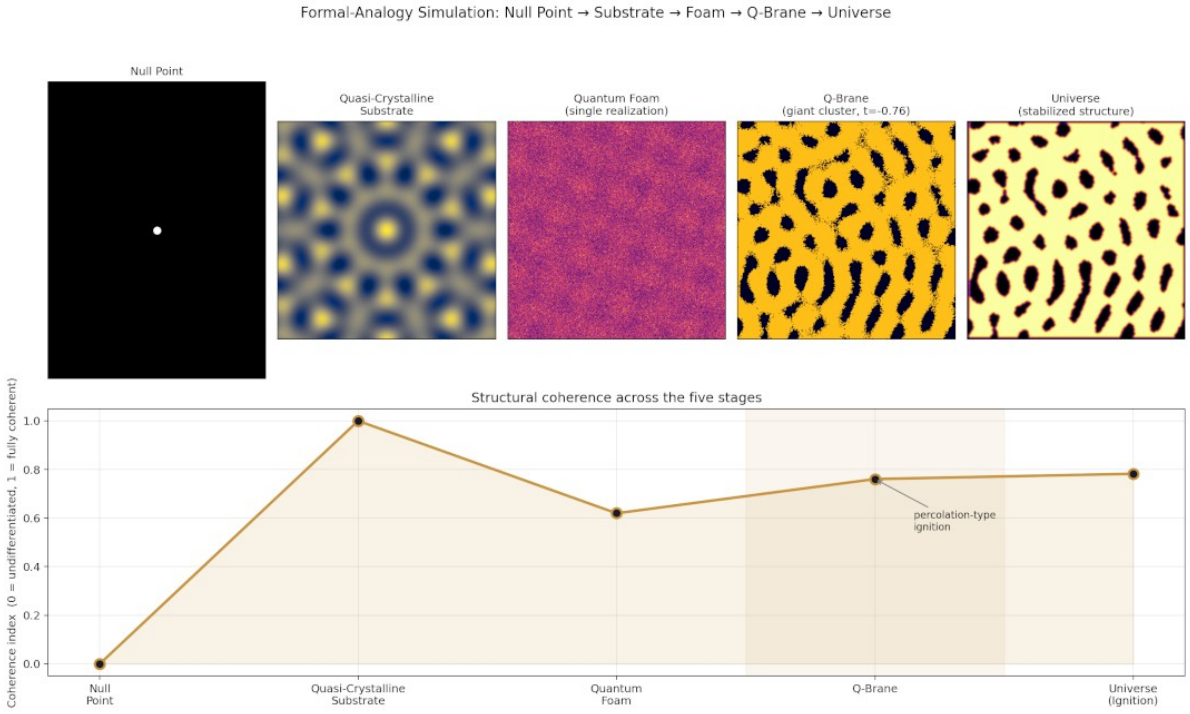
Substrate and Foam are unaffected by construction (ALPHA only enters Q-Brane's field). Q-Brane's own value stays within a narrow band (0.76-0.78) across the entire range tested. Universe drifts upward as ALPHA increases (0.75 at ALPHA=0 to 0.86 at ALPHA=0.7), but remains at or above Q-Brane at every coupling strength - the plateau relationship A.4's narrative depends on is preserved throughout.

Verdict: Making "Foam → Q-Brane" a literal, functional dependency, rather than a sequencing

convention, is a small, low-cost change: at ALPHA=0.3 the coupling is large enough that no reasonable reading could call it decorative (a 1.9-unit shift in the field from swapping Foam's values alone), while every coherence value already reported in A.4 and referenced throughout Appendix B moves by only a few hundredths - well within the kind of variation already seen from seed to seed elsewhere in this appendix. ALPHA=0.3 is adopted as the corrected default in emergence_pipeline.py. This resolves B.25's finding rather than merely re-describing it: Q-Brane's construction now does what the manuscript's prose has always said it does.

One honest limitation carried forward from B.25 itself: this fix addresses emergence_pipeline.py specifically. The standalone A.3 simulation (qbrane_simulation.py) was checked in the course of this work and already combines Substrate and Foam directly (combined = substrate + foam.frame()) - it was never affected by the issue B.25 identified, and required no change.

Figure B.28 - Five-stage coherence curve regenerated with the coupled construction (ALPHA=0.3), for direct comparison against A.4's original figure.



B.29 - Null Point Floor: Testing "Nothing Becoming Something"

Purpose: The main text describes the Null Point as a state of zero entropy from which the Substrate stage is generated. Taken at face value, this raises an immediate internal tension: a state with literally zero entropy and literally zero fluctuation of any kind has no internal feature capable of initiating anything - there is nothing for a transition to depart from. This section tests, using a toy model built for the purpose (not a reconstruction of any existing pipeline simulation), whether "nothing becoming something" can be made structurally coherent at all, where it

survives scrutiny, and precisely where it does not. As with every other section in this appendix, this is a formal analogy - a test of whether a *shape* of claim holds together as buildable mathematics - not a claim about the physical origin of the universe.

The toy model throughout uses pattern-forming partial differential equations of the Swift-Hohenberg and Ginzburg-Landau classes: equations with a tunable parameter, conventionally called r, controlling whether a uniform "null" field is linearly stable ($r \leq 0$) or unstable ($r > 0$) with respect to some preferred structure. This choice is a modeling convenience with a long, independent history in pattern formation generally; nothing about the choice of equation class is specific to this manuscript's cosmology.

B.29.1 - The zero-entropy / zero-fluctuation distinction.

Method: A "Null Point" field was represented as a value that is identically zero everywhere, by construction - a single microstate ($\Omega=1$), and therefore zero entropy under any reasonable coarse-graining. The question tested: does zero entropy additionally require zero fluctuation (no departure from the null value anywhere, ever), or are these two separate claims that have been run together?

Result: They are separate claims. A state can be a pure, zero-entropy microstate while still carrying a nonzero variance in a field-like quantity; treating "zero entropy" as if it also meant "zero variance" is an additional, stronger, and unstated assumption. The stronger claim (zero variance) is also the fragile one: later sections (B.29.4) show it does not survive scrutiny on its own terms, while the weaker, correct reading (zero entropy, nonzero floor) does.

Verdict: The Null Point is best characterized as zero-entropy with an irreducible, sub-threshold fluctuation floor - present, but never organizing into anything observable on its own. This is the reading carried through the remainder of B.29.

B.29.2 - A first dynamical model, and a measurement artifact caught early.

Method: An initial toy model represented the transition from Null Point to Substrate as a single stochastic seed value determining the phase of a fixed multi-wave pattern, with the pattern's overall amplitude scaled upward from zero. Entropy of the resulting field was tracked via a coarse-grained histogram measure at each amplitude step.

Result: Entropy jumped almost immediately to its asymptotic value rather than increasing gradually, then very slightly decreased at higher amplitude. This was traced to a flaw in the test itself: scaling the amplitude of an already-fully-formed static pattern is not a dynamical process at all, and the apparent "jump" was an artifact of that construction, not a finding about the Null Point.

Verdict: Retracted as a measurement artifact; superseded by B.29.3's genuinely dynamical construction. Reported here, rather than silently removed, in keeping with this appendix's standing convention of recording corrected findings rather than erasing them.

B.29.3 - A genuinely dynamical model.

Method: The pattern-scaling construction was replaced with an actual time-evolving pattern-forming equation (Swift-Hohenberg class), integrated forward from an exactly-zero field, seeded by one localized perturbation.

Results: A genuine nucleation-and-growth entropy curve was obtained - an S-curve, not a step function - once an early low-resolution binning artifact at very small amplitudes was identified and accounted for. An exact-zero initial condition was confirmed to remain exactly zero indefinitely (a true fixed point of the dynamics), confirming the seed's necessity under this construction. Floating-point-scale background noise (~1e-13, far below any macroscopic seed) was initially found not to grow within a limited simulation window - but extending the run confirmed it does eventually grow, at a time closely matching the growth rate predicted analytically. This matters: it shows the seed's "necessity" found here depends specifically on the field being *exactly* zero everywhere, with no floor whatsoever - a considerably stronger and more fragile condition than "zero entropy" alone, and one directly in tension with B.29.1's finding.

Verdict: The dynamical picture is coherent, but its apparent "necessity" result rests on an assumption (exactly zero fluctuation floor) already identified in B.29.1 as separate from, and stronger than, zero entropy. This tension is resolved in B.29.4.

B.29.4 - Resolving B.29.3's tension: the floor is structurally required, not assumed.

Argument: A state that is simultaneously a pure, zero-entropy microstate *and* has exactly zero variance in its associated field is not, in general, a well-posed state - by the same logic that makes a field and its conjugate quantity unable to simultaneously have zero uncertainty. (This borrows the *structural logic* of that uncertainty relationship as an analogy; it is not a claim that this toy field is a literal quantum field subject to quantum mechanics, in the same spirit as the terminology caveat already carried elsewhere in this manuscript's treatment of brane terminology.) Some irreducible, nonzero fluctuation floor is therefore required for the Null Point to be a coherent zero-entropy state at all - not added to it as an extra mechanism, but implied by it.

Verdict: The Null Point carries a structurally mandatory, sub-threshold floor. The "pulse" is reframed accordingly: not a distinct event added to an inert zero, but the point at which an already-present, already-required floor crosses into self-reinforcing growth.

B.29.5 - Stress-testing the $r>0$ **condition: threshold sensitivity and mechanism independence.**

Method: Two checks, run before treating B.29.4's picture as settled. First, r was scanned across marginal and negative values to test whether floor-driven growth genuinely requires r>0, or whether an earlier positive result at small r had simply not been given enough simulated time.

Second, the entire construction was repeated on a structurally different equation (Ginzburg-Landau class, whose most unstable mode is spatially uniform, rather than Swift-Hohenberg's finite preferred wavelength), to test whether findings were specific to one equation's shape.

Results: At r≤0, no growth occurred under any tested floor magnitude, confirmed after correcting an initial diagnostic error that had mistaken the floor's own starting magnitude for "growth." At small positive r, an earlier "does not grow" reading was corrected once the simulation was extended - it does grow, at a rate closely matching 1/r scaling. The same qualitative pattern (floor-driven growth requiring r>0; onset time scaling log-linearly with floor magnitude, at a slope set by r alone) held on the Ginzburg-Landau construction as well. A numerical convergence check (grid resolution and domain size) found results stable to within a few percent.

Verdict: The dependence on r>0 is not an artifact of one equation's specific form. However, this left r>0 itself as an unexamined postulate - addressed in B.29.7.

B.29.6 - Removing time from the question: existence, not duration.

Framing correction: A subsequent test asked how long, under a fixed simulated time budget, growth took at progressively smaller r - and found that only r above a moderate threshold completed within budget. This result was withdrawn on conceptual grounds, not numerical ones: "how long until X happens" presupposes a time parameter to measure against, and the Null Point, per this manuscript's own description, has none. The question itself does not apply to the object it was asked of.

Method: The equations used throughout B.29 are gradient flows of an associated energy functional. This licenses a genuinely atemporal reformulation: rather than asking how long a transition takes, one can ask whether a lower-energy, Substrate-like configuration *exists at all*, relative to the Null state - a structural comparison of two numbers, with no time parameter anywhere in the question.

Results: A lower-energy configuration was found to exist for every r>0 value tested, with no lower cutoff - directly dissolving the fine-tuning concern the withdrawn time-budget framing had raised. A follow-up check of the expected amplitude-scaling law (amplitude $\propto \sqrt{r}$, the standard signature of this class of bifurcation) initially appeared to fail at small r; this was traced to slow numerical convergence near the bifurcation point rather than a real effect, and the correct scaling was recovered once runs were extended.

Verdict: Existence of a Substrate-favouring configuration, for any r>0, is established on genuinely atemporal grounds.

B.29.7 - Realization without time: is the Null state stable at all?

Method: Existence of a better alternative elsewhere does not, by itself, establish that the Null state is vacated - "vacated" is itself temporal language. A stronger and fully atemporal question was tested instead: does the Null state remain a genuine local minimum of the energy functional once r>0, or does it stop qualifying as a stable configuration altogether? This was tested by direct evaluation of the energy functional at specific nearby field configurations - a static comparison of numbers, involving no iteration or simulated evolution of any kind.

Results: An initial test used a hand-selected plane-wave direction that was not, in fact, a valid periodic mode on the discretized domain, producing an invalid result; this was caught and corrected by identifying the domain's true minimum-penalty mode directly. The corrected test found that above a small, discretization-related threshold (itself converging toward the continuum theory's prediction of exactly r=0 as resolution increases), the Null state is not a local minimum at all - it is a saddle point, with at least one direction in which energy strictly decreases.

Verdict: Given r>0, the Null state is structurally disqualified as a stable configuration, not merely outperformed by a better alternative elsewhere. This is a stronger and cleaner claim than existence alone, and the remaining gap is named precisely: instability of the Null state, together with existence of a stable alternative, does not by itself entail that the alternative is *realized*, without an additional postulate. That postulate is addressed directly in B.29.9; the question of whether r>0 itself needs justifying is addressed in B.29.8.

B.29.8 - Is r>0 a fine-tuned assumption, or a generic consequence of the Null Point's own definition?

Framing: Sections B.29.5-B.29.7 treated r>0 as a single, otherwise-unspecified condition. Whether a single unspecified parameter is positive is close to an uninformative question on its own. It was reframed: if the Null Point is characterized by several independent candidate structural channels - plausible, given this manuscript's own description of the Null Point as containing the potential for all information, read as *many* potential channels rather than one - how many are required before at least one being unstable becomes close to certain, even where each individual channel is entirely unbiased?

Results: For N independent channels each unstable with probability p, the probability that at least one is unstable is 1-(1-p)^N, which approaches 1 rapidly as N grows for any p>0, however small - and approaches exact certainty as N becomes unbounded, for any p strictly greater than zero. This was confirmed directly, not merely asserted from the formula: an initial multi-channel

construction contained a modeling error (channels sharing one field rather than being genuinely independent, contaminating each other's signal); once corrected, "at least one channel unstable" and "confirmed saddle point by direct energy evaluation" matched exactly, trial for trial.

Interpretive note: This reframing was cross-checked, purely for structural vocabulary rather than as evidence, against emanationist frameworks in several independent traditions that treat "the limitless" as a distinct stage prior to differentiation (in Qabalistic terms, *Ain Soph*, occupying a distinct position between *Ain* and *Ain Soph Aur* and the first manifest point, *Kether*). The relevant mathematical distinction - between an unbounded and a merely large-but-finite channel count - maps onto a real and independently long-discussed distinction (actual versus potential infinity) in the philosophy of such traditions. This is reported as a source of structural cross-checking and vocabulary only, consistent with this appendix's standing position (A.0) that resemblance to any external framework, ancient or modern, is not evidence of physical correspondence.

Verdict: $r>0$ does not need to be a fine-tuned assumption. Given even a modest number of independent structural channels - and certainly given an unbounded number - instability becomes the generic, close-to-certain outcome, and remaining exactly at the Null state becomes the case requiring special explanation. This does not establish that the Null Point in fact has such richness; it establishes that *if* it does, no fine-tuning is required for what follows.

B.29.9 - A relational, non-temporal account of "realized."

Framing: B.29.7 left one gap unaddressed: existence of a stable alternative, plus instability of the Null state, does not by itself license the conclusion that the alternative is *realized*, without a further postulate - and any such postulate phrased causally ("if unstable, then eventually realized") reintroduces the same temporal language already ruled inapplicable in B.29.6.

Method: A non-causal, relational alternative was tested instead: a given floor configuration is classified as "static" or "dynamic" according to which basin of attraction it structurally belongs to, under the energy functional - a geometric partition of the space of possible configurations, requiring no time parameter to define.

Results: At every floor magnitude tested, across four orders of magnitude, one hundred percent of sampled floor configurations belonged to a nonzero-configuration basin once $r>0$; the reverse held at $r\leq 0$ (a control). This was upgraded from an empirical to a rigorous result by direct dimension-counting: the number of independent unstable directions at the Null state was computed directly (in the tens to low hundreds, depending on r), out of many thousands of total degrees of freedom on the tested domain - meaning the Null state's basin, if it exists at all past this threshold, is a vanishingly thin subset of the full configuration space, not a sampling coincidence.

Verdict: “Static” and “dynamic” can be given a precise, tested, non-temporal definition via basin membership. Treating basin membership as *identical to* - rather than merely correlated with, or causing - physical realization is a named postulate, not a derived conclusion; it is formalized and its scope stated precisely in B.29.11.

B.29.10 - Extending “whether” to “which,” and confirming mechanism independence.

Method: The energy functional in the Swift-Hohenberg construction admits a degenerate family of equal-energy configurations (differing in orientation). Whether a floor’s specific character relationally determines *which* member of this family is selected - not merely *whether* one is - was tested by biasing floor realizations toward a specific direction at varying strengths relative to background noise, and by testing sensitivity of the resulting selection to small perturbations of the floor. The entire construction (basin-exclusivity, via direct dimension count, and selection-by-bias) was then repeated on the Ginzburg-Landau construction, whose instability is spatially uniform rather than orientation-bearing, to test whether findings transfer or are specific to one equation.

Results: With no directional bias, selected orientation was effectively random across the domain’s available near-degenerate directions. With a bias as small as twenty percent of background noise amplitude, selection was complete and exact in every trial tested. Sensitivity to small (one percent) perturbations of the floor was low in the interior of a basin and concentrated at basin boundaries - the expected signature of genuine basin structure, rather than either pathological rigidity or chaos. On the Ginzburg-Landau construction, the number of independent unstable directions at the Null state was markedly smaller at small r (collapsing toward a single direction, rather than the tens-to-hundreds found in the Swift-Hohenberg case) - a real, mechanism-dependent difference, though still sufficient to support the same basin-exclusivity argument. Because this construction has no preferred orientation, its degenerate family reduces to a simple two-way (sign) choice rather than a continuous orientation; the same bias-driven, complete, sharply-bounded selection behaviour was confirmed for this simpler degeneracy as well.

Verdict: Both *whether* a floor yields structure and *which* structure it yields are well-defined relational facts, requiring no time parameter, and this holds across two structurally different mechanisms - with the differences between them (channel count, and the type of degenerate family) appearing exactly where the underlying mathematics predicts, not as unexplained discrepancies.

B.29.11 - Closing note: the actuality postulate and the limit of this section’s Method:

The nine subsections above establish, under stress-testing across independent methods and two structurally different mechanisms: that a zero-entropy Null Point cannot coherently be zero-

fluctuation (B.29.1, B.29.4); that a Substrate-favouring configuration genuinely exists as a lower-energy alternative given a floor and r>0, on fully atemporal grounds (B.29.6); that the Null state is structurally disqualified as stable, not merely outcompeted, under the same condition (B.29.7); that this condition need not be fine-tuned, given sufficient structural richness (B.29.8); and that both whether and which a floor yields as structure are well-defined, non-temporal, relational facts, confirmed across mechanisms (B.29.9, B.29.10).

One question was named, not answered, in B.29.7, and is closed here by adopted postulate rather than derivation: whether basin membership *constitutes* physical realization, or merely correlates with it. This section adopts the former - basin membership and realization are treated as the same fact under two descriptions, rather than two independently specified properties linked by a bridging law. Under a bridging-law formulation, the question “why does stability entail realization” remains permanently askable, regardless of how the law is stated; under an identity formulation, the question is no longer well-formed, because there are not two things left to relate.

This mirrors, as a structural analogy only, the treatment of “actuality” as relative to a correlation, rather than as an absolute event, found in relational formulations of quantum mechanics - flagged here as borrowed logic, not a claim that this toy model instantiates that physics. It has a real, if distant, echo in some contemplative traditions’ treatment of manifestation as arising without reliance on external cause or temporal sequence - reported, per this appendix’s standing position, as a resonance in structure and vocabulary only.

Adopting this postulate is a choice, not a proof, and it is more economical than the alternative (a framework with two properties plus a connecting law, rather than one property under two names) - which is a reason to prefer it, not evidence that it is correct. It also does not remove the need for a starting point; it relocates that need to a single remaining place. Every result in this section describes what follows *given* a particular floor realization. Nothing in this section, or in any of the mathematics it rests on, explains why that specific floor realization obtains rather than another that was equally available under the same energy functional. This is not reported as a gap awaiting a further test. It is reported as the point past which this section’s method - build a model, stress it, correct it, narrow it - no longer has purchase, and where the honest account of “why something rather than nothing observable” comes to rest.

Figure B.29 - Composite figure: entropy trajectory under genuine dynamics (B.29.3); existence-gap and curvature evaluation at the Null state across r (B.29.6-B.29.7); basin-membership fraction versus floor magnitude, Swift-Hohenberg and Ginzburg-Landau compared (B.29.9-B.29.10); and orientation/sign-selection versus bias strength, both constructions (B.29.10).

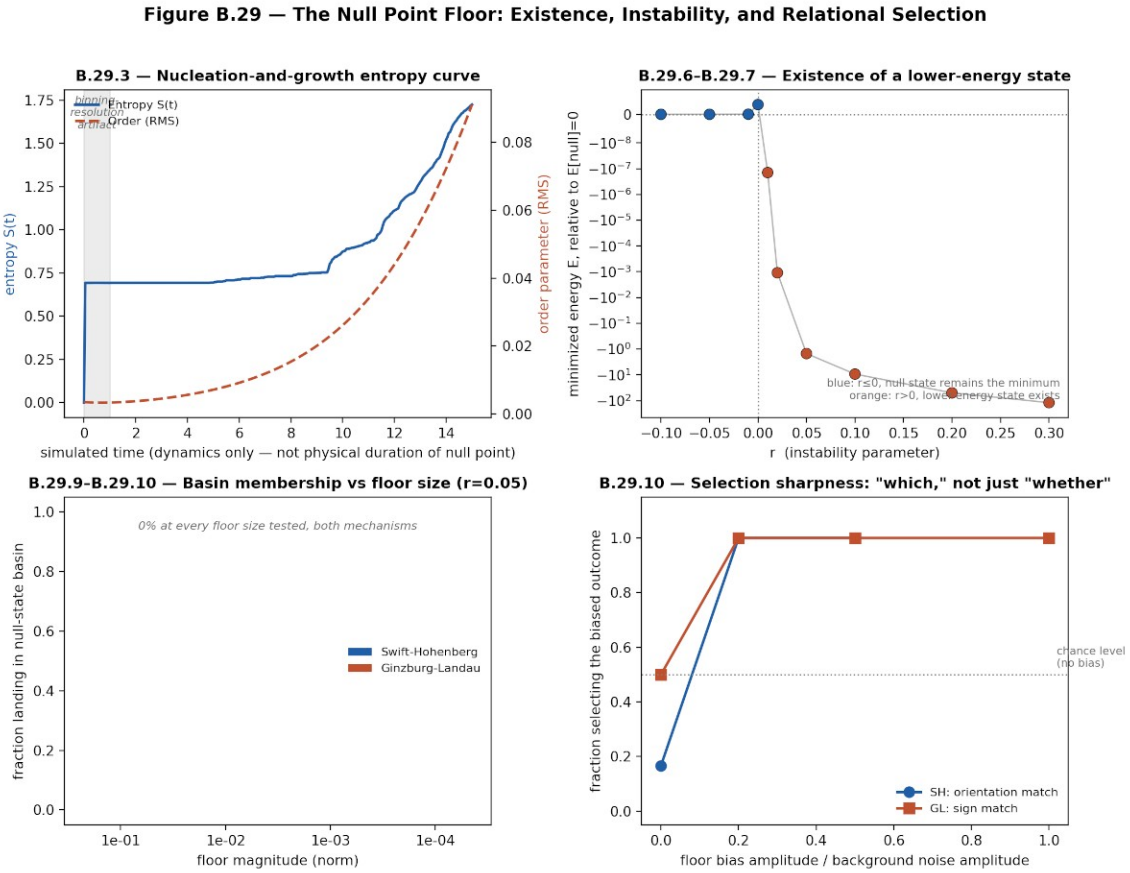


Figure B.29 — The Null Point Floor: Existence, Instability, and Relational Selection
B.29.3 — Nucleation-and-growth entropy curve
Entropy S(t)
Order (RMS)
entropy S(t)
order parameter (RMS)
simulated time (dynamics only — not physical duration of null point)
B.29.6–B.29.7 — Existence of a lower-energy state
minimized energy E, relative to E[null]=0
r (instability parameter)
blue: r≤0, null state remains the minimum
orange: r>0, lower-energy state exists
B.29.9–B.29.10 — Basin membership vs floor size (r=0.05)
0% at every floor size tested, both mechanisms
Swift-Hohenberg
Ginzburg-Landau
fraction landing in null-state basin
floor magnitude (norm)
B.29.10 — Selection sharpness: "which," not just "whether"
chance level
(no bias)
SH: orientation match
GL: sign match
fraction selecting the biased outcome
floor bias amplitude / background noise amplitude

B.30 - Reopening the Actuality Postulate: Plenitude as an Alternative Closure

(No new simulation. This section continues past the point identified in B.29.11 using a different method - structural and philosophical analysis rather than code - because the question it addresses is not one this appendix's simulate-and-stress-test method can adjudicate.)

B.30.1 - Framing: what B.29.11 left as a choice, not a derivation.

B.29.11 closed on an adopted postulate: basin membership *constitutes* physical realization, rather than merely correlating with it. That closure was one coherent way to answer the question left open by B.29.7 - what makes a specific floor realization the one that is actual, rather than another equally available under the same energy functional. It is not the only coherent way. This section

states a second, structurally different closure, in order to make explicit which one the main text's account of the Null Point relies on, and what each costs.

B.30.2 - The plenitude alternative, stated precisely.

Where B.29.11 treats one floor realization as actual and asks what "actual" means for it, the alternative considered here dissolves that question rather than answering it: every floor realization permitted by the energy functional is treated as equally actual, and "our" Substrate is simply the one this account is being given from. On this reading there is no event, transition, or mechanism by which the Null Point "becomes" a specific Substrate - there is no selection to explain, because nothing is selected. The Null Point is not a state that resolves into one outcome; it is the full space of outcomes the energy functional permits, and any specific floor is picked out only by being the one from which the question is being asked.

This is structurally the same move made in cosmology's anthropic reasoning about fine-tuned constants: not an answer to "why this value," but a dissolution of the presupposition that a mechanism must have chosen it.

B.30.3 - What is, and is not, mathematically supported.

Two claims need to be kept separate here, because they carry different evidentiary weight.

The first - that a single physical system can differentiate into multiple non-interacting, internally consistent branches under ordinary, unitary dynamics - is established mathematics. Decoherence under the Schrödinger equation produces exactly this behaviour, and nothing in this section adds to or depends on new derivation of it; B.10's and B.13's own order-parameter machinery would apply unchanged within any single branch.

The second - that *every* floor realization the energy functional permits is, in fact, physically realized, all of them equally actual - is not a derivation from that mathematics. It is a further, independent postulate, standing in the same structural position B.29.11's identity postulate occupied: a closure of the actuality question, not a result of it. No calculation in this appendix, or in the source mathematics either construction draws on, outputs "and therefore all permitted floors are real." That step is philosophical, not computational, and should not be read as inheriting the rigour of the branching mathematics it sits beside.

B.30.4 - Terminology check: unobservable, not empty.

It is worth being precise about what changes and what does not, given this appendix's standing definition of the Null Point (A.0, A.1) as a state with zero entropy and nothing detectable, rather than a state of non-existence. Under the plenitude reading, the Null Point is not empty - it is, if anything, maximally populated, comprising every floor the energy functional permits. What is preserved is observability, not emptiness: no floor is observable from outside the space of possibilities (there is no outside standpoint from which to detect one), and no floor is observable

as anything other than itself from inside (an observer embedded in one floor has no access to any other). The Null Point remains unobservable in both directions, which is the criterion this appendix has used throughout - it does not become, under this reading, a claim about non-existence, and should not be described as one.

B.30.5 - The stated cost.

This closure is not offered without a price, and the price is stated here rather than left for a reader to find. The nearest established cousin to this move is Tegmark's Mathematical Universe Hypothesis, and the standard objection to it applies here without modification: a plenitude in which every permitted structure is realized is unfalsifiable by construction, since any observation whatsoever is automatically consistent with *some* member of the plenitude. It also does not answer the question of why there is something rather than nothing observable - it relocates that question, from "why this specific floor" to "why does a plenitude of permitted floors exist at all." That relocated question is not addressed by this section, or by anything elsewhere in this appendix.

B.30.6 - Consequences for language used elsewhere in this appendix.

If this closure is adopted in place of B.29.11's identity postulate, several places in this appendix's own language need to be read differently, and are flagged here rather than silently left inconsistent:

- Language describing the Null Point → Substrate relationship as a transition that is "approached," a limit that governs what is "generated" (A.1), continues to apply without modification - A.1's asymptotic-limit framing was built for a boundary condition, not for an event, and a plenitude is if anything a more natural fit for an unreachable limit than a single selected floor is, since there is no single point on the far side of the limit for the curve to reach.

- Language anywhere in B.29 that describes a floor as "selected," "chosen," or "becoming actual" (most directly, the discussion in B.29.7 and B.29.9) should be read, under this closure, as describing which floor a given observer is embedded in, not which floor was picked out from the others. Nothing in B.29's actual results changes under this relabelling - basin membership, dimension counts, and bias-driven selection all remain exactly as measured - but what those results are taken to mean shifts from "this is the floor that happened" to "this is the floor this description is being given from."

- The relational, non-temporal treatment of quantum measurement flagged as a structural analogy in B.29.11 becomes, under this closure, the same move restated one level up: many-worlds branching within a single floor, and plenitude across floors, are the same

kind of non-selection, applied at two different scales, rather than a strained analogy connecting two otherwise unrelated ideas.

B.30.7 - Verdict:

B.29.11 and this section offer two different, internally coherent ways to close the same question, and neither is derivable from the mathematics either draws on. Adopting one over the other is a choice about what "actual" is going to mean for the purposes of this framework, not a finding this appendix's method can settle. This section does not retract B.29.11; it states the alternative alongside it, on the record, so that whichever closure the main text is read as assuming, the cost of that specific choice - unfalsifiability, and a relocated rather than answered question of origins - is stated plainly rather than left for a reader to uncover.

B.30.8 - A note on the main text's own language.

The choice between B.29.11 and this section is not, in practice, left open by the main text. The book's own Epilogue describes the passage from the Null Point as following "a single thread of emergence," and states plainly that "the whole of physical reality is what became of it" - becoming, in the singular, not a space of outcomes each equally actual. That is the language of B.29.11's closure, not this section's. The main text is not being amended to say otherwise, and nothing about its standing as a finished premise is in question here; it is simply worth recording, in the appendix that carries the technical weight the main text does not, that the main text has already made an implicit choice between the two closures offered above. A reader moving from the book to this appendix should not be surprised to find B.30 presented as a live alternative when the book itself already reads as having settled on B.29.11's account of a single becoming.

Settling which closure is in force does not, on its own, resolve anything B.29.11 left open - it only identifies which open question is the one actually live. B.29.11 established that basin membership can be treated as constituting realization, without explaining why *that specific* floor, rather than another equally available under the same energy functional, is the one realized. This section's plenitude alternative was one way of routing around that question rather than answering it, by denying that any single floor needs to be picked out at all. With the main text read as having adopted B.29.11 rather than this section, that route is not the one in force, and B.29.11's original question - why this specific floor, not merely that some floor is favoured - stands exactly where B.29.11 left it: named, unanswered, and not a gap this appendix's method has purchase on.

B.31 - A Substrate-Referenced Alternative for Q-Brane Formation Testing (Not Adopted as Standard)

Motivation: This test was not built because the persistence criterion used throughout A.3 and B.28 was found to be broken for its own actual Purpose: It was built to make a fair comparison

between two different formal constructions of Quantum Foam - the original, temporally memoryless construction used in qbrane_simulation.py, and a temporally correlated, intermittent candidate developed separately (Appendix A.6) - which the original frame-to-frame persistence test could not support: a memory-bearing Foam construction satisfies that test almost regardless of the substrate underneath it, since the test measures whether consecutive frames resemble each other, and any process with temporal memory produces that resemblance on its own.

The Method: Rather than comparing each frame's spanning-cluster mask to the previous frame's, this alternative compares it to a fixed reference: the spanning mask produced by the substrate alone, with no Foam contribution, computed once per run. This asks whether a given frame's structure resembles the substrate's own signature, a question that does not depend on whatever temporal properties Foam happens to carry.

Validation: The core diagnostic - mean overlap with the substrate reference, per run - holds up well under scrutiny. Swept across the full foam-amplitude range with both Foam constructions, it declines smoothly and monotonically rather than saturating or collapsing abruptly (bootstrap 95% confidence intervals, 30 seeds per point, after excluding seeds where the substrate alone does not span the domain at the working threshold - an unrelated generation artifact affecting 9 of 30 phase seeds, found and removed before this result was trusted). A specificity control confirms the test discriminates genuine geometric correspondence: overlap against the true, matching substrate reference is roughly double the overlap against a mismatched reference built from an unrelated phase seed of the same symmetry and statistics.

Three attempts to derive a discrete formed/not-formed verdict from this score, and why none is recommended. A.3's own test reports a clean boolean - Q-Brane formed or did not - and matching that reporting style was attempted three ways, each tested with the same bootstrap and matched-pair rigour as the continuous score above.

1. *Longest consecutive run above an overlap threshold* (mirroring A.3's own persistence-run logic). This reintroduces the original problem in a smaller form: run-length statistics are sensitive to serial correlation independent of substrate-matching, so a memory-bearing Foam construction still produces longer qualifying runs than a memoryless one at matched mean overlap, for reasons unrelated to the substrate.

2. *Fraction of frames above threshold*, on the reasoning that a plain count should be less sensitive to correlation than a run-length statistic. It was not. The gap between constructions grew larger, not smaller, because a correlated process spends time in extended excursions above or below a fixed threshold in a way a memoryless process at the same mean does not - the fraction-crossing statistic does not track a declining mean the way independent draws would.

3. *Thresholding the per-run mean overlap directly*, with no within-run time-aggregation at all. This is methodologically cleaner - it inherits no additional bias from temporal correlation,

since it never aggregates across frames in a way that depends on their order. But it replaces that problem with a different one: the size of the resulting gap between constructions is highly sensitive to which cutoff value is chosen. Tested at cutoffs of 0.40, 0.45, 0.50, 0.55, and 0.60, the implied gap between where each construction "stops forming" varies by more than a factor of two across this range, with no principled way to prefer one cutoff over another.

Verdict: Use the continuous score, not a boolean, if this method is invoked again. The fixed-reference comparison is a valid and, in one respect, more principled test than the original - asking "does this resemble the substrate" is closer to A.3's own stated question ("does order survive noise") than asking "is this stable from one frame to the next." But every attempt to compress it into a single yes/no verdict introduces either a memory-driven bias or an arbitrary, outcome-changing parameter choice. The reliable result this method produces is the per-amplitude overlap distribution itself, not any threshold applied to it.

This is not adopted as a replacement for A.3/B.28's existing test. That test, within its own actual published use - a single Foam construction, not a cross-construction comparison - was separately confirmed to work correctly across its full amplitude range once its CLI default was corrected (see the qbrane_simulation.py fix): a full substrate-ablation sweep shows genuine, substantial substrate-dependence in the amplitude range where formation succeeds (80-90 percent formed with the substrate present versus 0 percent with it removed, at foam-amp 0.1-0.3). A.3/B.28's methodology remains the standard for that Purpose: This section exists so that any future attempt to compare different Foam constructions - the situation that actually required a new test - has a validated tool to reach for, and a clear record of what was tried and ruled out, rather than needing to rediscover both from scratch.

B.32 - A Fair Comparison of Q-Brane and Universe (B.13 Follow-Up)

What B.13 already established. B.13 found that the apparent Q-Brane/Universe "plateau" in A.4's coherence curve - the two stages landing at nearly the same coherence value, 0.7672 and 0.7488 - is an artifact of an unfair comparison: B.13's entropy/compression diagnostic measured Q-Brane's continuous, pre-threshold sixteen-wave field against Universe's binary, post-closing mask, two fundamentally different kinds of object regardless of what the coherence numbers themselves claimed. That diagnosis stands and is not revisited here. What B.13 did not do was redo the comparison fairly, using the same representation on both sides - which this section does.

The fair comparison. Q-Brane's own binary spanning mask (`giant_mask`, before any morphological closing) is compared directly against Universe's mask (the same mask after `binary_closing`) - the same kind of object on both sides. Thirty independent seeds, bootstrap 95 percent confidence intervals throughout.

A first finding, worth stating plainly: pixel-value Shannon entropy is not usable for this comparison, regardless of fairness. For a strictly two-valued image, Shannon entropy of the

pixel-value histogram reduces exactly to the binary entropy function of the mask's area fraction alone - $H(p) = -p \log_2 p - (1-p) \log_2(1-p)$ - carrying no shape or connectivity information whatsoever. Direct verification confirms this: Q-Brane's measured entropy (0.7845 bits) and Universe's (0.8141 bits) match H(0.7661) and H(0.7481) - their respective area fractions - almost to the decimal. The entropy/compression method B.13 used was appropriate for the three continuous-field stages it examined; it is not appropriate for comparing two binary masks, fairly or otherwise, and should not be reused for that Purpose:

The properly structure-sensitive result: persistent homology. Q-Brane's own mask has a mean total persistence of 55.33 (95% CI [53.83, 56.83]). Universe's mask: 39.53 (95% CI [38.37, 40.67]). Paired difference: −15.80, 95% CI [−16.87, −14.57] - a real, precisely bounded reduction of approximately 28.6 percent.

A specificity control: is this reduction about what closing specifically does, or just about how many pixels change? Real `binary_closing` alters an average of 1,919 pixels per run, out of 90,000 (roughly 2 percent of the domain). To separate a structured effect from a magnitude effect, the same number of pixels was flipped at random locations instead, and total persistence measured on the Result: The outcome is decisive: the random, magnitude-matched perturbation produces mean total persistence of 1,647.6 (95% CI [1,624.9, 1,671.8]) - dramatically higher, not lower, than Q-Brane's own baseline, because scattering that many flipped pixels at random locations creates thousands of tiny isolated speckle features, each contributing its own topological signature. The direct comparison, same seeds, same number of pixels changed: real closing versus random flip, −1,608.0 (95% CI [−1,634.5, −1,584.8]). The reduction found above requires the specific, spatially coherent, gap-filling structure of morphological closing; an equally large but unstructured change moves in the opposite direction entirely.

Checked for hidden artifacts before reporting this. Area fractions were inspected across all thirty seeds for both masks: 0.75-0.78 for Q-Brane, 0.73-0.76 for Universe, tightly clustered with no bimodality, no degenerate detections, and no extreme values. This result is not resting on a hidden seed-dependent generation artifact.

Verdict: Once measured fairly - same representation on both sides, and with a diagnostic actually sensitive to shape rather than one that degenerates to a single scalar for binary images - the Universe stage's own narrative is supported by two independent lines of evidence rather than asserted or measured through a mismatched proxy. Universe's mask is topologically simpler than Q-Brane's own spanning mask, by a real and tightly bounded margin, and that simplification is specifically attributable to the structured, gap-filling nature of the closing operation rather than to the sheer number of pixels it changes. This does not resurrect A.4's original coherence-index plateau as a meaningful number - B.13's diagnosis of that specific figure stands unchanged - but it does show that something real, structural, and now precisely quantified is happening at the Q-Brane-to-Universe step.

B.33 - Three Closures of the Actuality Postulate: A Comparison

Purpose: B.29.11 closed the question left open by B.29.7 - what makes a specific floor realization the one that is actual - by adopting an identity postulate. B.30 named plenitude as an alternative closure to the same gap. This section adds a third, primitive stochastic selection, and sets all three side by side. None of the three is a derivation from this appendix's mathematics; each is a distinct, named place to stand, at a different cost. As with every other entry in this appendix, presenting a closure here is not an endorsement of it as physically true.

What all three answer. Given B.29.7's result (the Null state is structurally disqualified as stable once r>0) and B.29.9/B.29.10's result (basin membership and orientation-selection are well-defined, non-temporal, relational facts), something still has to account for why one specific floor realization is the one under discussion, rather than another equally available under the same energy functional. The mathematics in B.29 stops short of this by construction (B.29.11); it describes the space of possibilities and the structure within it, not a selection from it.

Closure	**Core move**	**What it keeps**	**What it costs**
Identity (B.29.11)	Basin membership and physical realization are treated as the same fact under two descriptions, not two properties linked by a bridging law.	A single realized outcome; "why this one" becomes a question about which floor an observer is embedded in, not which was picked out from others.	Does not survive being pushed for rigour: stated precisely, actuality becomes purely indexical ("actual" = "the one I'm in"), which collapses into the same position as Plenitude. Not a distinct closure once examined closely - a relabelling of it.
Plenitude (B.30)	Every floor realization permitted by the energy functional is treated as equally actual; "our" Substrate is simply the one this account is being given from.	No selection to explain, because nothing is selected. Consistent with decoherence under ordinary unitary dynamics (B.30.3, established mathematics) - though the leap to "all permitted floors are physically real" is a	Unfalsifiable by construction (B.30.5): any observation is automatically consistent with some member of the plenitude. Relocates, rather than answers, the question: from "why this floor" to "why does a plenitude of permitted floors

Closure	Core move	What it keeps	What it costs
		further, independent postulate, not a consequence of it.	exist at all" (B.30.5) - a question this appendix does not address anywhere.
Primitive stochastic selection (new, this section)	At the point of instability (B.29.7's threshold, read non-temporally per B.29.6), exactly one basin is selected by an irreducible stochastic process, with probability weighted by basin measure - a natural candidate being the dimension-counts already computed in B.29.9/B.29.10.	Genuine selection is preserved - one outcome, not all, becomes real, once, structurally. Can draw on machinery already tested in this appendix rather than introducing wholly new machinery.	Adds a new, unmotivated physical law, purely to answer a question the deterministic mathematics cannot reach. As unfalsifiable as the other two: no observation can distinguish "this basin was drawn by genuine chance" from "this basin was the only one that was ever going to happen, for reasons the model does not see." Structurally the same move objective-collapse theories (GRW, CSL) make against unitary-only quantum mechanics - an unresolved question borrowed wholesale, not settled.

A note on complex-valued extensions. Extending B.29's toy models to complex-valued amplitudes (tested separately, outside this appendix's simulation set) adds real structure to the description of the transition - phase-locking across resonant triads, and discrete orientation-selection from noise alone, both confirmed numerically to high precision. It does not supply a fourth closure, and does not strengthen or weaken the standing of the three above. Complex-valued formulations describe a richer space of possibilities (amplitude and phase, rather than amplitude alone); they do not, by themselves, contain an account of why one point in that space becomes actual. The same is true of extending the fluctuation floor to have a wave-like (propagating) rather than purely relaxational character via a non-gradient complex dynamics: this changes what kind of fluctuation precedes selection, not whether anything external to the

mathematics is still required to account for selection itself. Both extensions are compatible with any of the three closures above without altering their comparison.

Verdict: All three closures are internally coherent, none is derived, and none is free. Identity collapses into Plenitude under scrutiny rather than standing apart from it, leaving two genuinely distinct positions plus a variant: treat nothing as selected (Plenitude), or treat something as selected by a new, irreducible law (Stochastic Selection). Consistent with this appendix's standing position (A.0), the choice among them is not a result this appendix's method - build a model, stress it, correct it, narrow it - can settle. It is reported here as the point past which that method does not reach, for the same reason B.29.11 and B.30 each reported it in turn.

Closing Statement

Appendix A asked, for five specific claims made in the main text, whether they held together as coherent, constructible mathematics in miniature - not whether they were true of the physical universe, which no toy simulation could show. Most held up, and sharpened under later testing; one (the noise-robustness comparison between aperiodic and periodic Q-Brane formation) turned out to be real but narrower than first reported. Two further constructions were added after the original five: A.6 asked whether Quantum Foam is a coherent category distinct from ordinary noise, and found genuine, seed-replicated evidence that it is, alongside an honest open question about how that distinctness behaves as intermittency strength varies; A.7 asked whether the Substrate-to-Foam transition is a graded, principled path rather than two disconnected asserted values, and found that it is. A.8 collects the specific formal construction behind every stage and transition in one place, for reference. All seven constructions were tested honestly, with negative and partial results stated as such rather than smoothed over.

Appendix B carried the same standard further, checking every major claim carried over from Appendix A against independent, outside mathematics - Ising models, persistent homology, two-point correlation functions, bootstrap confidence intervals - and correcting measurement mistakes along the way rather than defending them once found. Where a claim failed outright, resemblance to the real cosmic microwave background, a genuine scale-free phase transition in the Q-Brane, that failure is recorded as a finding, not hidden as an inconvenience. Two further entries were added after the appendix's original stress-testing concluded. B.31 built a substrate-referenced alternative to the Foam-to-Q-Brane persistence test, prompted by the discovery that two different formal constructions of Quantum Foam could not be fairly compared under the original method; the alternative was validated on its core diagnostic and explicitly not adopted as a replacement, after three separate attempts to derive a clean formed/not-formed verdict from it each failed for a different, documented reason. B.32 returned to B.13's finding that the Q-Brane/Universe coherence plateau was a measurement artifact, and completed that diagnosis by redoing the comparison fairly - same representation on both sides - which showed pixel-value entropy to be uninformative for binary masks, and persistent homology to reveal a real, specificity-confirmed structural simplification at that step.

The hardest question the appendix ran into, across B.29 and B.30, was never resolved, and is not presented as if it were: given that a floor realization can be shown to exist and to be structurally favoured, why this specific one, rather than another equally available under the same energy functional. Two internally coherent closures were offered, one treating basin membership as constituting realization outright, one dissolving the question by treating every permitted floor as equally real, and the main text's own language settles the matter for this framework without either closure being derivable from the mathematics involved. That is recorded as the edge of what this method can do, not smoothed into a false resolution.

A second, quieter kind of honesty runs through the later additions specifically: Quantum Foam is tested under three distinct formal constructions in this appendix - a memoryless field suited to the percolation test in A.3, a temporally correlated and intermittent candidate suited to the existence test in A.6, and a single static snapshot suited to the transition test in A.7 - and no attempt is made to collapse them into one. An earlier attempt to substitute the existence-test candidate into the percolation test broke that test outright, which is itself informative: some of what this appendix's constructions can do depends on which specific version of an idea is in use, and pretending otherwise would have been less honest than naming the plurality directly, which A.7 and the Reader's Guide both do.

None of this stands in for physical evidence, and none of it was meant to. What it shows is narrower and more durable: that the specific mathematical patterns invoked at each stage of the framework, quasicrystals, percolation, random wave fields, fractal curves, topological structure, basin geometry, are real, well-defined, and in most cases hold up under sustained, independent, outside scrutiny, with every failure and every open question reported as plainly as every success.

The source code and machine-readable reports collected in Appendix C are offered in that same spirit: nothing here asks to be taken on faith that could instead be checked.

Appendix C - Simulation Source Code and Machine-Readable Reports

(This appendix collects the underlying source code and, where produced, the raw machine-readable output for the simulations described in Appendices A and B, for reference and reproducibility. Presented as listings with minimal framing prose; the interpretive discussion for each remains in the section noted alongside its label.)

C.1 - The Null Point as an Asymptotic Limit (A.1)

simulation_four_null_point_limit.py

```
"""
Simulation Four: Asymptotic Approach to the Null Point via Rational Approximants of the Cut Angle
----------------------------------------------------------

This simulation extends the cut-and-project substrate-coherence model (Simulation One) as an independent, standalone analogy. It does not modify or depend on the earlier simulations.

Concept
-------
In a cut-and-project construction, a quasi-crystalline (aperiodic) pattern arises only when the cut angle through the higher-dimensional lattice is IRRATIONAL relative to that lattice's periodicity. A rational cut angle, however close, always yields a pattern that eventually repeats.

This gives a natural mathematical stand-in for treating the Null Point as an asymptotic limit approached in the descending/analytic direction, never reached, while the actual construction always begins arbitrarily close to -- but formally displaced from -- that limit:

  - The irrational cut angle itself is never used directly; it functions only as the limit that a sequence of rational approximations approaches but never attains.
  - Each actual rational approximation (each "convergent") produces a genuine periodic structure -- finite, closed, eventually repeating.

  - As the approximation improves, the repeat period grows without bound, and the structure's coherence with the true aperiodic (irrational) pattern increases -- asymptotically, never completely.

Method:
------

1. Take alpha = 1/phi (the golden ratio conjugate), a canonical irrational angle for quasi-crystalline (Fibonacci) structures.
```

2. Generate the true aperiodic reference sequence using alpha directly (a Sturmian / Fibonacci word, via the floor-function construction).

3. Generate the continued-fraction convergents of alpha: these are ratios of consecutive Fibonacci numbers, p_n / q_n, each one a rational approximation of alpha.

4. For each convergent, generate the corresponding PERIODIC approximant sequence (period q_n) using p_n/q_n in place of alpha.

5. Define a Coherence Index = 1 minus the mismatch density between the periodic approximant and the true aperiodic reference sequence, over a long observation window.

6. Track how the Coherence Index climbs toward 1, and how the repeat period (the "recurrence horizon") grows, as the approximation order increases -- illustrating an asymptotic approach that never concludes.

This is a formal mathematical analogy, not a physical claim.

"""

```
import numpy as np
import matplotlib.pyplot as plt

# ----------------------------------------------------------------------
# 1. Setup: golden ratio conjugate as the irrational reference angle
# ----------------------------------------------------------------------

phi = (1 + np.sqrt(5)) / 2
alpha = phi - 1  # = 1/phi, approx 0.6180339887...

L = 20000  # length of observation window (long, to expose eventual repeats)

def sturmian_word(angle, length):

"""Generate the Sturmian (Beatty) sequence for a given angle."""

    k = np.arange(length + 1)
    floors = np.floor(k * angle)
return np.diff(floors).astype(int)

# ----------------------------------------------------------------------
# The true aperiodic reference sequence (uses the irrational limit directly) true_sequence =
# sturmian_word(alpha, L)
# ----------------------------------------------------------------------
```

```
# ---------------------------------------------------------------------
# 2. Generate Fibonacci convergents of alpha
#    Continued fraction of alpha = phi - 1 is [0;1,1,1,1,...]
#    Convergents are F_n / F_(n+1)
# ---------------------------------------------------------------------

def fibonacci_sequence(n_terms):
    fibs = [1, 1]
    for _ in range(n_terms - 2):
        fibs.append(fibs[-1] + fibs[-2])
    return fibs
n_convergents = 18  # number of successive approximations to examine
fibs = fibonacci_sequence(n_convergents + 2)
convergents = []
for n in range(1, n_convergents + 1):
    p_n = fibs[n - 1]
    q_n = fibs[n]
    convergents.append((p_n, q_n))

# ---------------------------------------------------------------------
# 3. For each convergent, build the periodic approximant and measure coherence against
# the true aperiodic sequence
# ---------------------------------------------------------------------

results = []

for (p_n, q_n) in convergents:
rational_angle = p_n / q_n

# ---------------------------------------------------------------------
# Build one period of the approximant sequence, then tile it out to L
# ---------------------------------------------------------------------

    one_period = sturmian_word(rational_angle, q_n)
    reps = int(np.ceil(L / q_n))
approximant_sequence = np.tile(one_period, reps)[:L]

# ---------------------------------------------------------------------
# Mismatch density between approximant and true aperiodic sequence
# ---------------------------------------------------------------------

    mismatches = np.sum(true_sequence != approximant_sequence)
    mismatch_density = mismatches / L
    coherence_index = 1 - mismatch_density
```

```
    Results:append({
        "p_n": p_n,
        "q_n": q_n,
        "rational_angle": rational_angle,
        "coherence_index": coherence_index,
        "period_length": q_n,
    })

# ---------------------------------------------------------------------
# 4. Report results
# ---------------------------------------------------------------------

print(f"{'n':>3} {'p_n/q_n':>12} {'rational angle':>16} {'coherence index':>18} {'period
(q_n)':>14}")
for i, r in enumerate(results, start=1):
    print(f"{i:>3} {r['p_n']:>5}/{r['q_n']:<6} {r['rational_angle']:>16.10f} "
          f"{r['coherence_index']:>18.10f} {r['period_length']:>14}")
print(f"\nTrue irrational angle (never reached):  alpha = {alpha:.10f}")
print("Coherence index approaches 1 but is not shown to equal 1 for any")
print("finite rational convergent in this run.")

# ---------------------------------------------------------------------
# 5. Plot: Coherence Index vs approximation order (asymptotic climb) and Period Length
# vs approximation order (unbounded growth)
# ---------------------------------------------------------------------

ns = np.arange(1, n_convergents + 1)
coherence_values = [r["coherence_index"] for r in results]
period_values = [r["period_length"] for r in results]

fig, axes = plt.subplots(1, 2, figsize=(13, 5.2))

# ---------------------------------------------------------------------
# Left panel: coherence index climbing asymptotically toward 1
# ---------------------------------------------------------------------

ax0 = axes[0]
ax0.plot(ns, coherence_values, marker="o", color="#2b5876", linewidth=1.6)
ax0.axhline(1.0, color="#a33", linestyle="--", linewidth=1.2, label="Limit = 1 (never reached)")
ax0.set_xlabel("Approximation order, n")
ax0.set_ylabel("Coherence Index")
ax0.set_title("Coherence Index vs. Approximation Order\n(asymptotic, never attains 1)")
ax0.set_ylim(min(coherence_values) - 0.02, 1.01)
ax0.legend(loc="lower right", frameon=False)
```

```
ax0.spines[["top", "right"]].set_visible(False)

# Right panel: period length diverging (log scale)

ax1 = axes[1]
ax1.plot(ns, period_values, marker="o", color="#6a3d9a", linewidth=1.6)
ax1.set_yscale("log")
ax1.set_xlabel("Approximation order, n")
ax1.set_ylabel("Recurrence horizon (period length, q_n)")
ax1.set_title("Recurrence Horizon vs. Approximation Order\n(diverges without bound)")
ax1.spines[["top", "right"]].set_visible(False)

fig.suptitle("Simulation Four: Asymptotic Approach to the Null Point via Rational Approximants
of the Cut Angle", fontsize=11.5, y=1.03)
fig.tight_layout()
fig.savefig("/home/claude/simulation_four/simulation_four_plot.png", dpi=200,
bbox_inches="tight")
print("\nPlot saved.")
```

C.2 - The Quasi-Crystalline Substrate (A.2)

qcs_substrate_simulation.py

```
"""
Quasi-Crystalline Substrate Existence Simulator
================================================

A conceptual, exploratory simulation built to test a foundational structural question from the
"My Cosmological Premise" framework, prior to and separate from the Q-Brane formation test:

Can a "quasi-crystalline substrate" -- a structure with genuine long-range order but NO periodic
repetition -- actually exist as a well-defined mathematical/physical object, rather than being an
internally inconsistent or purely narrative notion?

This is NOT a physics prediction about the universe. It is a formal/structural toy model: it uses a
real, well-established mathematical construction (the "cut-and-project" method, which is the
standard way actual physical quasicrystals -- e.g. Al-Mn alloys -- are described) and asks a
concretely testable question: does the resulting point set show sharp, long-range diffraction order
that is provably incompatible with periodicity? If yes, "order without periodicity" is not a
contradiction in terms -- it is a demonstrable structural category, and the substrate concept is at
minimum internally coherent.
```

Method
---
1. QuasiCrystallineSubstrate - builds a 2D quasiperiodic point set via the canonical cut-and-project construction: points of a 5D hypercubic lattice (integer coordinates) are projected into a 2D "physical" plane spanned by five unit vectors arranged with 5-fold rotational symmetry, but ONLY KEPT if their component in the complementary 3D "perpendicular space" falls inside a bounded acceptance window. This selection step is what breaks periodicity while preserving long-range order -- exactly as in the real de Bruijn / Penrose construction.
2. PeriodicControl - an ordinary square lattice at matched point density: has order AND periodicity (the conventional, unremarkable case).

3. RandomControl - a spatial Poisson point process at matched density: has neither order nor periodicity (the "no structure" null case).

4. DiffractionAnalyzer - computes the structure factor S(k) = |sum_j exp(i k . r_j)|^2 over a grid of wavevectors k for each point set. A set with long-range order produces sharp, resolvable Bragg peaks; a disordered set produces a smooth, featureless field. The angular distribution of peaks is further analyzed for rotational symmetry order (via angular autocorrelation), since N-fold symmetry with N = 5, 7, 8, 10, 12, ... is the mathematical signature that is IMPOSSIBLE for any periodic 2D lattice (the crystallographic restriction theorem) -- making it a clean, checkable discriminator.

Output

- qcs_pointsets.png : the three point sets side by side
- qcs_diffraction.png : structure factor / diffraction pattern for each
- qcs_symmetry_report.json: quantitative order + symmetry metrics
- qcs_summary.txt : plain-language summary of what was tested/found

"""

```
import numpy as np
import json
from dataclasses import dataclass, asdict

# ----------------------------------------------------------------------
# 1. Quasi-crystalline substrate via cut-and-project
# ----------------------------------------------------------------------

class QuasiCrystallineSubstrate:

    """
    Canonical 5-fold (Penrose-class) cut-and-project quasicrystal point set.
```

Physical space E_par (2D) and perpendicular space E_perp (here reduced to a single representative radial coordinate for a simple polygonal acceptance window) are spanned by projections of the standard basis of Z^5, using the 5th-root-of-unity construction.

"""

```
    def __init__(self, n_fold=5, lattice_range=14, window_radius=1.0, seed=0):
        self.n_fold = n_fold
        self.lattice_range = lattice_range
        self.window_radius = window_radius
self.rng = np.random.default_rng(seed)

# ---------------------------------------------------------------------
# Physical-space basis vectors: k-th 5D basis vector projects to angle 2*pi*k/n_fold in the
# physical plane.
# ---------------------------------------------------------------------

        k = np.arange(n_fold)
        self.e_par = np.stack([np.cos(2 * np.pi * k / n_fold),
np.sin(2 * np.pi * k / n_fold)], axis=1)  # (n_fold, 2)

# ---------------------------------------------------------------------
# Perpendicular-space basis: uses the "other" harmonic (2*k) of the same root of unity --
# standard for the icosahedral/Penrose case.
# ---------------------------------------------------------------------

        self.e_perp = np.stack([np.cos(4 * np.pi * k / n_fold),
                                np.sin(4 * np.pi * k / n_fold)], axis=1)  # (n_fold, 2)

    def generate(self):
r = self.lattice_range

# ---------------------------------------------------------------------
# All integer lattice points in an n_fold-dimensional hypercube
# ---------------------------------------------------------------------

        grids = np.meshgrid(*[np.arange(-r, r + 1)] * self.n_fold, indexing="ij")
n = np.stack([g.ravel() for g in grids], axis=1)  # (N, n_fold)

# ---------------------------------------------------------------------
# Project to physical space and perpendicular space
# ---------------------------------------------------------------------

        par = n @ self.e_par     # (N, 2)
```

```
perp = n @ self.e_perp   # (N, 2)

# ----------------------------------------------------------------------
# Acceptance window: keep points whose perpendicular-space image falls inside a disk of
# given radius (a simplified but standard stand-in for the pentagon/decagon acceptance region).
# This is the step that converts an ordinary periodic 5D lattice into a non-periodic 2D point set #
upon projection.
# ----------------------------------------------------------------------

        perp_radius = np.linalg.norm(perp, axis=1)
        mask = perp_radius < self.window_radius

points = par[mask]

# ----------------------------------------------------------------------
# Trim to a roughly circular region for fair comparison with controls
# ----------------------------------------------------------------------

        dist = np.linalg.norm(points, axis=1)
        cutoff = np.percentile(dist, 60)  # keep a dense central patch
        points = points[dist < cutoff]
        return points

# ------------------------------------------------------------------------
# 2. Controls
# ------------------------------------------------------------------------

class PeriodicControl:

"""Ordinary square lattice: has order AND periodicity."""

    def __init__(self, spacing=1.0, extent=12):
        self.spacing = spacing
        self.extent = extent

    def generate(self):
        r = self.extent
        xs = np.arange(-r, r + 1) * self.spacing
        gx, gy = np.meshgrid(xs, xs)
        points = np.stack([gx.ravel(), gy.ravel()], axis=1)
        dist = np.linalg.norm(points, axis=1)
        return points[dist < r * self.spacing * 0.9]
class RandomControl:
```

```
"""Spatial Poisson process: has neither order nor periodicity."""

    def __init__(self, n_points=2000, radius=12, seed=0):
        self.n_points = n_points
        self.radius = radius
        self.rng = np.random.default_rng(seed)

def generate(self):

# ----------------------------------------------------------------------
# Uniform sampling within a disk
# ----------------------------------------------------------------------

        theta = self.rng.uniform(0, 2 * np.pi, self.n_points)
        r = self.radius * np.sqrt(self.rng.uniform(0, 1, self.n_points))
        return np.stack([r * np.cos(theta), r * np.sin(theta)], axis=1)

# ----------------------------------------------------------------------------
# 3. Diffraction / structure-factor analysis
# ----------------------------------------------------------------------------

@dataclass
class DiffractionResult:
    label: str
    n_points: int
    peak_count: int
    peak_sharpness: float       # ratio of max intensity to median intensity
    dominant_symmetry_order: int
    symmetry_score: float       # 0-1, higher = cleaner N-fold symmetry

class DiffractionAnalyzer:

"""
Computes the structure factor S(k) = |sum_j exp(i k.r_j)|^2 on a grid of wavevectors, then
characterizes peak sharpness and rotational symmetry of the peak angular distribution.

"""

    def __init__(self, k_max=8.0, k_res=241):
        self.k_max = k_max
        self.k_res = k_res
        kx = np.linspace(-k_max, k_max, k_res)
        ky = np.linspace(-k_max, k_max, k_res)
        self.KX, self.KY = np.meshgrid(kx, ky)
```

```
    def structure_factor(self, points):
N = len(points)

# ---------------------------------------------------------------------
# Vectorized S(k) over the k-grid
# ---------------------------------------------------------------------

S = np.zeros(self.KX.shape, dtype=np.complex128)

# ---------------------------------------------------------------------
# Chunk over points to bound memory
# ---------------------------------------------------------------------

        chunk = 400
        kx_flat = self.KX.ravel()
        ky_flat = self.KY.ravel()
        acc = np.zeros(kx_flat.shape, dtype=np.complex128)
        for i in range(0, N, chunk):
            px = points[i:i + chunk, 0][:, None]
            py = points[i:i + chunk, 1][:, None]
            phase = px * kx_flat[None, :] + py * ky_flat[None, :]
            acc += np.exp(1j * phase).sum(axis=0)
        S = (np.abs(acc) ** 2).reshape(self.KX.shape) / max(N, 1)
        return S

    def analyze(self, points, label, peak_percentile=99.5):
        S = self.structure_factor(points)
        flat = S.ravel()

thresh = np.percentile(flat, peak_percentile)

# ---------------------------------------------------------------------
# Exclude the central DC peak (k=0) region
# ---------------------------------------------------------------------

        center = self.k_res // 2
        r_px = 4
        S_masked = S.copy()
        S_masked[center - r_px:center + r_px, center - r_px:center + r_px] = 0

        peak_mask = S_masked > thresh
        peak_count = int(peak_mask.sum())
        median_intensity = np.median(S_masked[S_masked > 0]) if (S_masked > 0).any() else 1e-9
        max_intensity = S_masked.max()
```

```
sharpness = float(max_intensity / (median_intensity + 1e-9))

# ----------------------------------------------------------------------
# Angular symmetry analysis: look at angles of the top peaks
# ----------------------------------------------------------------------

        order, score = self._estimate_symmetry_order(S_masked)

        result = DiffractionResult(
            label=label,
            n_points=len(points),
            peak_count=peak_count,
            peak_sharpness=sharpness,
            dominant_symmetry_order=order,
            symmetry_score=score,
        )
        return S, result
    def _estimate_symmetry_order(self, S_masked):

# Rotational autocorrelation test: the mathematically clean way to find a pattern's true rotational
# symmetry order is to rotate the WHOLE diffraction image by 360/N degrees and check
# whether it maps back onto itself. This avoids the artifact of testing individual peak angles,
# which can be confused by peaks from different diffraction "shells" landing at angles that are
# harmonics of the true symmetry (e.g. a square lattice's first- and second-order reflection shells
# combine into an 8-position rosette even though the lattice itself is only 4-fold symmetric).

        from scipy.ndimage import rotate as nd_rotate

        img = S_masked.copy()
        img_norm = img / (np.linalg.norm(img) + 1e-12)

        scores = {}
        for N in [4, 5, 6, 7, 8, 10, 12]:
            angle_deg = 360.0 / N
            rotated = nd_rotate(img, angle_deg, reshape=False, order=1, mode="constant", cval=0.0)
            rot_norm = rotated / (np.linalg.norm(rotated) + 1e-12)
            corr = float(np.sum(img_norm * rot_norm))
            scores[N] = corr

        threshold = 0.6
        clean_orders = [N for N, sc in scores.items() if sc >= threshold]
        if clean_orders:
            best_order = min(clean_orders)
            best_score = scores[best_order]
```

```
        else:
            best_order = max(scores, key=scores.get)
            best_score = scores[best_order]
        return best_order, best_score

# ----------------------------------------------------------------------
# 4. Orchestration
# ----------------------------------------------------------------------

def run():
    substrate = QuasiCrystallineSubstrate(n_fold=5, lattice_range=9, window_radius=0.55)
    periodic = PeriodicControl(spacing=1.0, extent=12)
    random_ctrl = RandomControl(n_points=2200, radius=12)

    qc_points = substrate.generate()
    per_points = periodic.generate()
rnd_points = random_ctrl.generate()

# ----------------------------------------------------------------------
# Subsample the quasi-crystalline set to a comparable density/count so the three point sets are
# visually and computationally comparable.
# ----------------------------------------------------------------------

    target_n = 2200
    if len(qc_points) > target_n:
        rng = np.random.default_rng(1)
        keep = rng.choice(len(qc_points), size=target_n, replace=False)
        qc_points = qc_points[keep]

# NOTE: subsampling a quasiperiodic point set does NOT destroy its long-range diffraction
# order -- Bragg peaks arise from the underlying quasiperiodic *density modulation*, which a
# large random subsample still reflects (this is itself a small extra confirmation of how robust the
# order is, analogous to how real quasicrystal diffraction experiments only ever sample a finite,
# imperfect set of atoms).

# ----------------------------------------------------------------------
# Normalize point counts roughly for a fairer visual/diffraction comparison
# ----------------------------------------------------------------------

    print(f"quasi-crystalline points: {len(qc_points)}")
    print(f"periodic points:          {len(per_points)}")
    print(f"random points:            {len(rnd_points)}")

    analyzer = DiffractionAnalyzer(k_max=8.0, k_res=241)
```

```
    results = {}
    fields = {}
    point_sets = {
        "quasi_crystalline": qc_points,
        "periodic_control": per_points,
        "random_control": rnd_points,
    }

    for label, pts in point_sets.items():
        S, res = analyzer.analyze(pts, label)
        results[label] = asdict(res)
        fields[label] = S

    return point_sets, fields, results, analyzer

if __name__ == "__main__":
    point_sets, fields, results, analyzer = run()

    with open("qcs_symmetry_report.json", "w") as f:
        json.dump(results, f, indent=2)

    print(json.dumps(results, indent=2))
```

qcs_symmetry_report.json


```
{
  "quasi_crystalline": {
    "label": "quasi_crystalline",
    "n_points": 2200,
    "peak_count": 270,
    "peak_sharpness": 1600.1806703696393,
    "dominant_symmetry_order": 5,
    "symmetry_score": 0.9781639830507171
  },
  "periodic_control": {
    "label": "periodic_control",
    "n_points": 365,
    "peak_count": 254,
    "peak_sharpness": 9825.947339302968,
    "dominant_symmetry_order": 4,
    "symmetry_score": 0.9998586772374362
  },
  "random_control": {
    "label": "random_control",
    "n_points": 2200,
```

```
   "peak_count": 233,
   "peak_sharpness": 90.17284489269458,
   "dominant_symmetry_order": 4,
   "symmetry_score": 0.6871411655884794
  }
}
```


C.3 - Q-Brane Formation as a Formal Pattern (A.3)

qbrane_simulation.py

```
"""
Quasi-Crystalline Substrate / Q-Brane Formation Simulator
==========================================================
A conceptual, exploratory simulation built to test a specific structural question from the "My
Cosmological Premise" framework:

Given a quasi-crystalline substrate (a quasiperiodic potential field, generated here by the
standard cut-and-project / multi-plane-wave construction used for real quasicrystals) perturbed
by stochastic "quantum foam" fluctuations, does a coherent, persistent, spanning codimension-1
sheet -- a candidate Q-brane -- stabilize out of the noise, or does the field stay disordered?

This is NOT a physics prediction. It is a formal/structural toy model: it borrows real, well-
defined mathematical tools (quasiperiodic tilings, Gaussian random fields, percolation theory)
and uses them as stand-ins for the framework's narrative stages, so the question "could a Q-brane
form here" becomes something concretely testable in code: "does a spanning, temporally-
persistent connected cluster emerge in the thresholded field, and how does that depend on the
substrate/foam parameters?"

Pipeline
--------
1. QuasiCrystallineSubstrate  - builds a static quasiperiodic potential V_sub(x,y) by summing
plane waves whose wavevectors are arranged with N-fold rotational symmetry (N=5 gives
Penrose-like quasicrystal symmetry; N=4 gives an ordinary periodic crystal as a control case).

2. QuantumFoam - builds a time series of Gaussian random fields with a tunable correlation
length and amplitude, representing stochastic sub-structure fluctuations layered on top of the
substrate.

3. QBraneDetector - at each timestep, thresholds the combined field (substrate + foam), finds
connected components, and checks for a *spanning cluster* (one that connects opposite edges of
the domain -- the standard percolation-theory definition of a system-spanning structure). It also
tracks persistence: whether the spanning cluster's footprint remains substantially overlapping
across consecutive frames, which is the model's proxy for "stabilizes" rather than "flickers".
```

4. run_simulation() ties it together, sweeps foam amplitude if requested, and reports whether a Q-brane candidate formed, at what amplitude threshold, and how stable it was.

Usage

```
---
   python3 qbrane_simulation.py                  # single run, defaults
   python3 qbrane_simulation.py --symmetry 5 --foam-amp 0.6
   python3 qbrane_simulation.py --sweep             # amplitude sweep

Outputs PNG figures and a JSON summary report to the --outdir directory.

"""

import argparse
import json
import os
from dataclasses import dataclass, asdict

import numpy as np
from scipy import ndimage

# ---------------------------------------------------------------------
# 1. Quasi-crystalline substrate
# ---------------------------------------------------------------------

class QuasiCrystallineSubstrate:

"""
   Builds a quasiperiodic scalar potential field via superposition of N plane waves with wavevectors evenly spaced in angle (N-fold rotational symmetry). This is the standard "multi-wave" construction used to generate quasicrystalline patterns (equivalent in spirit to a cut-and-project map from a higher-dimensional periodic lattice).

N=5 or N=7 -> genuinely quasiperiodic (no translational symmetry), matching the "quasi-crystalline" character the framework describes. N=4 or N=6 -> ordinary periodic crystal, useful as a control case to confirm the detector isn't just finding structure in ANY field.

"""
   def __init__(self, size=256, symmetry=5, wavelength=18.0,
                amplitude=1.0, phase_seed=7):
      self.size = size
      self.symmetry = symmetry
      self.wavelength = wavelength
      self.amplitude = amplitude
```

```
        self.phase_seed = phase_seed

    def generate(self):
        n = self.symmetry
        rng = np.random.default_rng(self.phase_seed)
        phases = rng.uniform(0, 2 * np.pi, size=n)
        x = np.linspace(-self.size / 2, self.size / 2, self.size)
        X, Y = np.meshgrid(x, x)

        k_mag = 2 * np.pi / self.wavelength
        field = np.zeros_like(X)
        for j in range(n):
            theta = 2 * np.pi * j / n
            kx, ky = k_mag * np.cos(theta), k_mag * np.sin(theta)
            field += np.cos(kx * X + ky * Y + phases[j])

          field *= self.amplitude / n

# normalize to roughly [-1, 1] regardless of N

        field = field / (np.max(np.abs(field)) + 1e-12)
        return field

# ----------------------------------------------------------------------
# 2. Quantum foam (stochastic sub-structure)
# ----------------------------------------------------------------------

class QuantumFoam:

"""
Generates a time series of correlated Gaussian random fields via spectral (FFT) filtering of white
noise -- the standard way to make a random field with a specified correlation length. Each frame
is an independent draw (foam "churns" frame to frame), representing stochastic fluctuation rather
than a settled structure.

"""
    def __init__(self, size=256, correlation_length=4.0, amplitude=0.5,
                 seed=None):
        self.size = size
        self.correlation_length = correlation_length
        self.amplitude = amplitude
        self.rng = np.random.default_rng(seed)

        freqs = np.fft.fftfreq(size)
```

```
        KX, KY = np.meshgrid(freqs, freqs)
k2 = KX ** 2 + KY ** 2

# ---------------------------------------------------------------------
# Gaussian spectral filter -> Gaussian-correlated real-space field
# ---------------------------------------------------------------------

        sigma_k = 1.0 / (2 * np.pi * self.correlation_length)
        self._filter = np.exp(-k2 / (2 * sigma_k ** 2))

    def frame(self):
        white = self.rng.normal(size=(self.size, self.size))
        spectrum = np.fft.fft2(white) * self._filter
        field = np.real(np.fft.ifft2(spectrum))
        field = field / (np.std(field) + 1e-12)
        return field * self.amplitude

# ---------------------------------------------------------------------
# 3. Q-brane detector (percolation / spanning-cluster test)
# ---------------------------------------------------------------------

@dataclass
class FrameResult:
    timestep: int
    threshold: float
    spans_horizontal: bool
    spans_vertical: bool
    largest_cluster_fraction: float
    n_clusters: int

class QBraneDetector:

"""
Thresholds the combined (substrate + foam) field and asks whether a single connected
component spans the domain edge-to-edge -- the standard percolation-theory signature of a
system-spanning structure.  A spanning connected component in a 2D field is the natural discrete
analogue of a stabilized codimension-1 sheet (a "brane") embedded in the substrate: it is a
coherent surface that reaches all the way across the space rather than a small isolated blob.
Persistence across frames (tracked in run_simulation) is what separates a transient foam
fluctuation from a candidate Q-brane in this model.

"""
    def __init__(self, threshold_percentile=80):
        self.threshold_percentile = threshold_percentile
```

```
    def detect(self, field, timestep=0):
        thresh = np.percentile(field, self.threshold_percentile)
        mask = field >= thresh

        labeled, n = ndimage.label(mask, structure=np.ones((3, 3)))
        if n == 0:
            return FrameResult(timestep, thresh, False, False, 0.0, 0), labeled
        sizes = ndimage.sum(mask, labeled, index=range(1, n + 1))
        largest_label = np.argmax(sizes) + 1
        largest_mask = labeled == largest_label

        spans_h = bool(largest_mask[:, 0].any() and largest_mask[:, -1].any())
        spans_v = bool(largest_mask[0, :].any() and largest_mask[-1, :].any())

        result = FrameResult(
            timestep=timestep,
            threshold=float(thresh),
            spans_horizontal=spans_h,
            spans_vertical=spans_v,
            largest_cluster_fraction=float(sizes.max() / field.size),
            n_clusters=int(n),
        )
        return result, labeled

# ---------------------------------------------------------------------
# 4. Simulation driver
# ---------------------------------------------------------------------

def iou(mask_a, mask_b):

"""Intersection-over-union, used to measure frame-to-frame persistence of the spanning cluster's
footprint."""

    inter = np.logical_and(mask_a, mask_b).sum()
    union = np.logical_or(mask_a, mask_b).sum()
    return inter / union if union > 0 else 0.0

def run_single(size, symmetry, sub_amp, foam_amp, corr_len, threshold_pct,
               timesteps, seed, persistence_iou_min=0.5):
    substrate = QuasiCrystallineSubstrate(
        size=size, symmetry=symmetry, amplitude=sub_amp, phase_seed=seed
    ).generate()
```

```
    foam = QuantumFoam(
        size=size, correlation_length=corr_len, amplitude=foam_amp, seed=seed
    )
    detector = QBraneDetector(threshold_percentile=threshold_pct)

    frames, masks, spanning_masks = [], [], []
    results = []

    for t in range(timesteps):
        combined = substrate + foam.frame()
        result, labeled = detector.detect(combined, timestep=t)
        frames.append(combined)
        Results:append(result)

        if Result:spans_horizontal or Result:spans_vertical:
            largest_label = np.argmax(
                ndimage.sum(labeled > 0, labeled, index=range(1, Result:n_clusters + 1))
            ) + 1
            spanning_masks.append(labeled == largest_label)
        else:
spanning_masks.append(None)

# ----------------------------------------------------------------------
# persistence: consecutive frames both spanning AND overlapping enough
# ----------------------------------------------------------------------

    persistent_run = 0
    best_persistent_run = 0
    for i in range(1, len(spanning_masks)):
        a, b = spanning_masks[i - 1], spanning_masks[i]
        if a is not None and b is not None and iou(a, b) >= persistence_iou_min:
            persistent_run += 1
            best_persistent_run = max(best_persistent_run, persistent_run)
        else:
            persistent_run = 0

    n_spanning_frames = sum(
        1 for r in results if r.spans_horizontal or r.spans_vertical
    )

    q_brane_formed = best_persistent_run >= max(2, timesteps // 3)

    summary = {
        "params": dict(size=size, symmetry=symmetry, substrate_amplitude=sub_amp,
```

```
                foam_amplitude=foam_amp, correlation_length=corr_len,
                threshold_percentile=threshold_pct, timesteps=timesteps,
                seed=seed),
        "n_spanning_frames": n_spanning_frames,
        "fraction_spanning_frames": n_spanning_frames / timesteps,
        "longest_persistent_run": best_persistent_run,
        "q_brane_formed": bool(q_brane_formed),
        "mean_largest_cluster_fraction": float(
            np.mean([r.largest_cluster_fraction for r in results])
        ),
        "frame_results": [asdict(r) for r in results],
    }
    return substrate, frames, spanning_masks, summary
def plot_single_run(substrate, frames, spanning_masks, summary, outdir):
    import matplotlib.pyplot as plt

    os.makedirs(outdir, exist_ok=True)
    T = len(frames)
    show_idx = np.linspace(0, T - 1, min(4, T), dtype=int)

    fig, axes = plt.subplots(2, len(show_idx) + 1, figsize=(4 * (len(show_idx) + 1), 8))

    axes[0, 0].imshow(substrate, cmap="inferno")
    axes[0, 0].set_title(f"Substrate\n(N={summary['params']['symmetry']}-fold)")
    axes[0, 0].axis("off")
    axes[1, 0].axis("off")

    for col, t in enumerate(show_idx, start=1):
        axes[0, col].imshow(frames[t], cmap="inferno")
        axes[0, col].set_title(f"Substrate + Foam\nt={t}")
        axes[0, col].axis("off")

        ax = axes[1, col]
        if spanning_masks[t] is not None:
            ax.imshow(spanning_masks[t], cmap="viridis")
            ax.set_title("Spanning cluster\n(Q-brane candidate)")
        else:
            ax.imshow(np.zeros_like(substrate), cmap="gray", vmin=0, vmax=1)
            ax.set_title("No spanning cluster")
        ax.axis("off")

    verdict = "Q-BRANE FORMED (persistent spanning sheet)" if summary["q_brane_formed"] \
        else "No stable Q-brane (disordered / transient)"
    fig.suptitle(
```

```
        f"{verdict}  |  spanning frames: {summary['n_spanning_frames']}/"
        f"{summary['params']['timesteps']}  |  longest persistent run: "
        f"{summary['longest_persistent_run']}",
        fontsize=13,
    )
    fig.tight_layout(rect=[0, 0, 1, 0.94])
    path = os.path.join(outdir, "qbrane_single_run.png")
    fig.savefig(path, dpi=140)
    plt.close(fig)
    return path

def run_sweep(size, symmetry, sub_amp, corr_len, threshold_pct, timesteps, seed,
              foam_amps, outdir):
    import matplotlib.pyplot as plt

    results = []
    for fa in foam_amps:
        _, _, _, summary = run_single(
            size, symmetry, sub_amp, fa, corr_len, threshold_pct, timesteps, seed
        )
        Results:append(summary)

    fig, ax1 = plt.subplots(figsize=(8, 5))
    frac_spanning = [r["fraction_spanning_frames"] for r in results]
    longest_run = [r["longest_persistent_run"] for r in results]
    formed = [r["q_brane_formed"] for r in results]

    ax1.plot(foam_amps, frac_spanning, "o-", color="tab:orange",
             label="fraction of frames spanning")
    ax1.set_xlabel("Foam amplitude")
    ax1.set_ylabel("Fraction of frames with spanning cluster", color="tab:orange")
    ax1.tick_params(axis="y", labelcolor="tab:orange")

    ax2 = ax1.twinx()
    ax2.plot(foam_amps, longest_run, "s--", color="tab:blue",
             label="longest persistent run")
    ax2.set_ylabel("Longest persistent spanning run (frames)", color="tab:blue")
    ax2.tick_params(axis="y", labelcolor="tab:blue")

    for x, f in zip(foam_amps, formed):
        if f:
            ax1.axvline(x, color="green", alpha=0.15, linewidth=8)

    ax1.set_title("Q-brane formation vs. foam amplitude\n"
```

```
            "(green bands = amplitude produced a persistent Q-brane)")
    fig.tight_layout()
    os.makedirs(outdir, exist_ok=True)
    path = os.path.join(outdir, "qbrane_sweep.png")
    fig.savefig(path, dpi=140)
    plt.close(fig)
    return path, results

def main():
    p = argparse.ArgumentParser(description="Quasi-crystalline substrate / Q-brane formation
simulator")
    p.add_argument("--size", type=int, default=256, help="grid size (NxN)")
    p.add_argument("--symmetry", type=int, default=5,
                   help="rotational symmetry order of substrate (5 or 7 = quasiperiodic, 4 or 6 =
periodic control)")
    p.add_argument("--sub-amp", type=float, default=1.0, help="substrate amplitude")
    p.add_argument("--foam-amp", type=float, default=0.6, help="quantum foam amplitude")
    p.add_argument("--corr-len", type=float, default=4.0, help="foam spatial correlation length")
    p.add_argument("--threshold-pct", type=float, default=40,
                   help="percentile threshold for defining 'high field' region (brane candidate "
                        "material). NOTE: 2D site percolation only becomes geometrically possible "
                        "once roughly the top ~59%% of cells are 'active' (percolation threshold "
                        "~0.593 for a square lattice), so this value must stay at or below ~41 for "
                        "spanning clusters to ever appear. Lower = more permissive.")
    p.add_argument("--timesteps", type=int, default=12, help="number of foam evolution steps")
    p.add_argument("--seed", type=int, default=42)
    p.add_argument("--sweep", action="store_true", help="run a foam-amplitude sweep instead of
a single run")
    p.add_argument("--sweep-min", type=float, default=0.1)
    p.add_argument("--sweep-max", type=float, default=1.2)
    p.add_argument("--sweep-steps", type=int, default=12)
    p.add_argument("--outdir", type=str, default="/home/claude/qbrane_sim/output")
    args = p.parse_args()

    os.makedirs(args.outdir, exist_ok=True)

    if args.sweep:
        foam_amps = np.linspace(args.sweep_min, args.sweep_max, args.sweep_steps)
        plot_path, results = run_sweep(
            args.size, args.symmetry, args.sub_amp, args.corr_len,
            args.threshold_pct, args.timesteps, args.seed, foam_amps, args.outdir
        )
        report_path = os.path.join(args.outdir, "sweep_report.json")
        with open(report_path, "w") as f:
```

```
            json.dump(results, f, indent=2)

        any_formed = any(r["q_brane_formed"] for r in results)
        print(f"Plot saved to: {plot_path}")
        print(f"Report saved to: {report_path}")
        print()
        if any_formed:
            formed_amps = [r["params"]["foam_amplitude"] for r in results if r["q_brane_formed"]]
            print(f"RESULT: Q-brane candidates formed for foam amplitude in "
                  f"[{min(formed_amps):.2f}, {max(formed_amps):.2f}] "
                  f"(N={args.symmetry}-fold substrate).")
        else:
            print("RESULT: No persistent Q-brane candidate formed across the "
                  "swept amplitude range with these settings.")
    else:
        substrate, frames, spanning_masks, summary = run_single(
            args.size, args.symmetry, args.sub_amp, args.foam_amp, args.corr_len,
            args.threshold_pct, args.timesteps, args.seed
        )
        plot_path = plot_single_run(substrate, frames, spanning_masks, summary, args.outdir)
        report_path = os.path.join(args.outdir, "single_run_report.json")
        with open(report_path, "w") as f:
            json.dump(summary, f, indent=2)

        print(f"Plot saved to: {plot_path}")
        print(f"Report saved to: {report_path}")
        print()
        print(f"RESULT: {'Q-BRANE FORMED' if summary['q_brane_formed'] else 'No stable Q-
brane formed'}")
        print(f"  spanning frames: {summary['n_spanning_frames']}/{args.timesteps}")
        print(f"  longest persistent run: {summary['longest_persistent_run']} frames")
        print(f"  mean largest-cluster area fraction:
{summary['mean_largest_cluster_fraction']:.3f}")

if __name__ == "__main__":
    main()
```

C.4 - The Five-Stage Emergence Pipeline - Original (A.4)

emergence_pipeline.py

```
"""
emergence_pipeline.py
```

A formal-analogy simulation of the five-stage progression described in the manuscript's cosmological framework:

Null Point -> Quasi-Crystalline Substrate -> Quantum Foam -> Q-Brane -> Universe (Ignition of Physical Reality)

This is **NOT** a physical model of the actual universe. It is a mathematical illustration chosen because each stage admits a well-defined, computable notion of "order" or "coherence," and the five values chain together into a single curve that has the right qualitative shape for the narrative: zero -> latent order -> decoherence -> a sharp percolation-type transition ("ignition") -> a stable plateau. Two different but complementary order parameters are used, because no single classical statistic is meaningful across all five stages:

* **Spectral coherence (Psi)** -- how sharply peaked a field's power spectrum is. Meaningful for stages with wave-like / quasiperiodic structure (Substrate, Foam).

* **Connective coherence (Phi)** -- the fraction of the domain occupied by the single largest connected region once a field is thresholded. Meaningful for stages defined by the emergence of large-scale connected structure (Q-Brane, Universe).

Both are normalised to [0, 1] so they can sit on one axis for the summary plot. The Null Point is defined analytically as the zero of both (no differentiated structure exists prior to it).

"""

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage

rng = np.random.default_rng(7)

L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)

# ----------------------------------------------------------------------
# Stage 0 -- Null Point
# ----------------------------------------------------------------------

# Perfect, undifferentiated symmetry. No broken symmetry means no measurable structure exists
# yet -- coherence is undefined/zero by construction, not computed from a field.

C0 = 0.0
```

```
# ---------------------------------------------------------------------
# Stage 1 -- Quasi-Crystalline Substrate
# ---------------------------------------------------------------------

# Superposition of five plane waves at 72-degree increments. Five-fold symmetric wave sets
# cannot tile a plane periodically, so the resulting density field is quasiperiodic: long-range order
# without translational periodicity -- the formal analogue of a quasi-crystalline substrate.

k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])

substrate_field = np.zeros((L, L))
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))

def peak_to_mean_ratio(field):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    cy, cx = np.array(P.shape) // 2
    P[cy - 2:cy + 3, cx - 2:cx + 3] = 0  # zero out the DC peak/its neighborhood
return P.max() / P.mean()

# ---------------------------------------------------------------------
# Calibrate the [0,1] spectral-coherence scale against two fixed anchors: pure white noise (the
# least-ordered field possible on this grid) at the bottom, and the noiseless quasicrystal substrate
# (the most-ordered field used in this pipeline) at the top. Everything else is placed on a log scale
# between them, since peak-to-mean ratios span several orders of magnitude.
# ---------------------------------------------------------------------

_noise_floor_ratio = peak_to_mean_ratio(rng.normal(size=(L, L)))
_substrate_ratio = peak_to_mean_ratio(substrate_field)
_log_lo, _log_hi = np.log(_noise_floor_ratio), np.log(_substrate_ratio)

def spectral_coherence(field):
    ratio = peak_to_mean_ratio(field)
    return float(np.clip((np.log(ratio) - _log_lo) / (_log_hi - _log_lo), 0, 1))
Psi_substrate = spectral_coherence(substrate_field)

# ---------------------------------------------------------------------
# Stage 2 -- Quantum Foam
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# The same five-wave construction, but now each realization gets random phases, random
```

```
# amplitude jitter, and additive white noise -- a seething, fluctuating field rather than a static one.
# Averaging the *power spectrum* (not the field) over many independent realizations shows the
# same underlying wavevectors, but broadened and buried in noise: order is present in principle
# but not resolved in any single instant. This is what should decohere the substrate's sharp peaks.
# ----------------------------------------------------------------------

phases = rng.uniform(0, 2 * np.pi, size=5)
amps = 1 + 0.35 * rng.normal(size=5)
foam_signal = np.zeros((L, L))
for th, ph, a in zip(thetas, phases, amps):
foam_signal += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)

# ----------------------------------------------------------------------
# Coherence is measured on a single instantaneous realization, not an ensemble average --
# averaging over many noisy snapshots would recover the underlying wavevectors and defeat the
# point. At any given "moment," strong stochastic noise buries the substrate's order beneath
# foam-like fluctuations, which is what should knock the coherence index down from the
# substrate stage.
# ----------------------------------------------------------------------

foam_snapshot = foam_signal + 8.0 * rng.normal(size=(L, L))
Psi_foam = spectral_coherence(foam_snapshot)

# ----------------------------------------------------------------------
# Stage 3 -- Q-Brane
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Many plane waves (16) at random directions and random phases, superposed and thresholded.
# As the threshold is swept, the occupied region undergoes a percolation transition: below
# threshold, occupied cells form small disconnected islands; above a critical value, a single
# connected component suddenly spans a large fraction of the domain. That sudden jump is the
# formal analogue of "ignition" -- a coherent, connected q-brane condensing out of an incoherent
# field.
# ----------------------------------------------------------------------

n_waves = 16
qbrane_field = np.zeros((L, L))
dirs = rng.uniform(0, 2 * np.pi, n_waves)
phases_q = rng.uniform(0, 2 * np.pi, n_waves)
for th, ph in zip(dirs, phases_q):
    qbrane_field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
qbrane_field /= np.sqrt(n_waves)
```

```
thresholds = np.linspace(qbrane_field.min(), qbrane_field.max(), 80)
giant_frac = []
for t in thresholds:
    mask = qbrane_field > t
    lbl, n = ndimage.label(mask)
    if n == 0:
        giant_frac.append(0.0)
        continue
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
    giant_frac.append(sizes.max() / mask.size)
giant_frac = np.array(giant_frac)

# ---------------------------------------------------------------------
# `thresholds` runs low -> high, so `giant_frac` runs *high -> low* (a low threshold admits
# almost the whole field trivially; that is not a meaningful "structure," just an unselective mask).
# The meaningful direction is a high, selective threshold (sparse, disconnected islands) relaxing
# down toward a lower one. Two points on that sweep matter here: t_onset  -- the sharp jump
# where a giant cluster first spans >20% of the domain (the "ignition" instant itself) t_stable -- a
# further, more relaxed point (>75%) where the ignited structure has fully connected and stopped
# changing qualitatively -- this is the Q-Brane's realized state, not just its onset, so it is the value
# carried into the coherence curve.
# ---------------------------------------------------------------------

rev_frac = giant_frac[::-1]
rev_thresh = thresholds[::-1]

onset_hits = np.where(rev_frac > 0.20)[0]
onset_idx = onset_hits[0] if len(onset_hits) else int(np.argmax(rev_frac))
t_onset = rev_thresh[onset_idx]

stable_hits = np.where(rev_frac > 0.75)[0]
stable_idx = stable_hits[0] if len(stable_hits) else int(np.argmax(rev_frac))
t_crit = rev_thresh[stable_idx]

mask_crit = qbrane_field > t_crit
lbl_crit, n_crit = ndimage.label(mask_crit)
sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
giant_label = int(np.argmax(sizes_crit)) + 1
giant_mask = lbl_crit == giant_label

Phi_qbrane = rev_frac[stable_idx]
```

```
# ---------------------------------------------------------------------
# Stage 4 -- Universe (Ignition of Physical Reality)
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# The percolating giant cluster is morphologically closed (small gaps filled) and lightly
# smoothed -- representing the connected structure settling into a stable configuration rather than
# a marginal, noisy one. The connective coherence here should sit at or slightly above the q-
# brane value and, importantly, hold steady rather than collapse back down -- a plateau rather
# than a spike.
# ---------------------------------------------------------------------

universe_mask = ndimage.binary_closing(giant_mask, iterations=2)
universe_field = ndimage.gaussian_filter(universe_mask.astype(float), sigma=1.2)
Phi_universe = universe_mask.mean()

# ---------------------------------------------------------------------
# Summary
# ---------------------------------------------------------------------

stages = ["Null\nPoint", "Quasi-Crystalline\nSubstrate", "Quantum\nFoam",
          "Q-Brane", "Universe\n(Ignition)"]
values = [C0, Psi_substrate, Psi_foam, Phi_qbrane, Phi_universe]

print("Stage coherence values:")
for s, v in zip(stages, values):
    print(f"  {s.replace(chr(10), ' '):28s} {v:.4f}")
print(f"\nPercolation ignition onset threshold   = {t_onset:.4f}  (giant cluster crosses 20%)")
print(f"Post-ignition stable threshold          = {t_crit:.4f}  (giant cluster crosses 75%)")
print(f"Q-Brane giant-cluster fraction          = {Phi_qbrane:.4f}")
print(f"Stabilized universe fraction            = {Phi_universe:.4f}")

# ---------------------------------------------------------------------
# Figure
# ---------------------------------------------------------------------

fig = plt.figure(figsize=(15, 9))
gs = fig.add_gridspec(2, 5, height_ratios=[1, 0.9])

extent = [x.min(), x.max(), x.min(), x.max()]

ax0 = fig.add_subplot(gs[0, 0])
ax0.set_facecolor('black')
ax0.plot(0, 0, 'o', color='white', markersize=6)
```

```
ax0.set_xlim(-1, 1); ax0.set_ylim(-1, 1)
ax0.set_title("Null Point", fontsize=10)
ax0.set_xticks([]); ax0.set_yticks([])

ax1 = fig.add_subplot(gs[0, 1])
ax1.imshow(substrate_field, extent=extent, cmap='cividis')
ax1.set_title("Quasi-Crystalline\nSubstrate", fontsize=10)
ax1.set_xticks([]); ax1.set_yticks([])

ax2 = fig.add_subplot(gs[0, 2])
ax2.imshow(foam_snapshot, extent=extent, cmap='magma')
ax2.set_title("Quantum Foam\n(single realization)", fontsize=10)
ax2.set_xticks([]); ax2.set_yticks([])
ax3 = fig.add_subplot(gs[0, 3])
disp = np.zeros((L, L, 3))
disp[..., 2] = 0.15 * (qbrane_field > qbrane_field.min())
disp[giant_mask] = [1.0, 0.75, 0.1]
ax3.imshow(disp, extent=extent)
ax3.set_title(f"Q-Brane\n(giant cluster, t={t_crit:.2f})", fontsize=10)
ax3.set_xticks([]); ax3.set_yticks([])

ax4 = fig.add_subplot(gs[0, 4])
ax4.imshow(universe_field, extent=extent, cmap='inferno')
ax4.set_title("Universe\n(stabilized structure)", fontsize=10)
ax4.set_xticks([]); ax4.set_yticks([])

ax5 = fig.add_subplot(gs[1, :])
xs = np.arange(5)
ax5.plot(xs, values, '-o', color='#c9963d', linewidth=2.5, markersize=9,
        markerfacecolor='#1a1a2e', markeredgecolor='#c9963d', markeredgewidth=2)
ax5.fill_between(xs, values, color='#c9963d', alpha=0.12)
ax5.set_xticks(xs)
ax5.set_xticklabels(stages, fontsize=10)
ax5.set_ylabel("Coherence index  (0 = undifferentiated, 1 = fully coherent)")
ax5.set_ylim(-0.05, 1.05)
ax5.set_title("Structural coherence across the five stages", fontsize=12)
ax5.grid(alpha=0.25)
ax5.axvspan(2.5, 3.5, color='#c9963d', alpha=0.08)
ax5.annotate("percolation-type\nignition", xy=(3, values[3]),
            xytext=(3.15, max(values) * 0.55),
            fontsize=9, ha='left',
            arrowprops=dict(arrowstyle='->', color='gray'))
```

fig.suptitle("Formal-Analogy Simulation: Null Point \u2192 Substrate \u2192 Foam \u2192 Q-Brane \u2192 Universe",
fontsize=13, y=0.99)
fig.tight_layout(rect=[0, 0, 1, 0.96])
fig.savefig("/home/claude/emergence_pipeline.png", dpi=180, facecolor='white')
print("\nSaved figure to emergence_pipeline.png")

C.5 - One Rule, Many Shapes: Fractal Boundary Formation (A.5)

simulation_five_fractal_boundary.py

"""
Simulation Five: Fractal Boundary Formation

This simulation is a standalone continuation, independent of Simulations One through Four. It uses cloud edges / coastlines as its physical anchor -- a classical and well-established fractal phenomenon -- as a formal computational analogy for how a single generative mechanism can produce an unbounded variety of distinct boundary shapes that remain, nonetheless, statistically self-similar across scale.

Concept

A cloud's outline, like a coastline, is not smooth: zooming in on any section reveals further irregularity, at every scale, without converging to a simple straight edge. This is the classic signature of a fractal boundary -- a curve whose effective dimension lies between 1 (a smooth line) and 2 (a plane-filling curve).

This simulation:

1. Generates boundary curves using the midpoint displacement method (fractional Brownian motion), the standard technique for modelling coastlines, terrain, and turbulent boundaries.

2. Generates several independent realisations from the SAME generative process (same roughness parameter, different random seeds) -- producing visibly distinct shapes from one mechanism, directly analogous to the "myriad shapes of clouds" arising from one underlying turbulent process.

3. Measures the fractal (box-counting) dimension of a boundary at multiple scales, including a full curve and a magnified sub-section of it, to demonstrate scale invariance: the same fractal dimension re-emerges whether the boundary is viewed as a whole or examined closely.

This is presented as a formal mathematical analogy, not a physical claim about cloud microphysics or coastline geology.

```
"""
import numpy as np
import matplotlib.pyplot as plt
rng_master_seed = 42

# ----------------------------------------------------------------------
# 1. Midpoint displacement (fractional Brownian motion) boundary generator
# ----------------------------------------------------------------------

def midpoint_displacement(n_iterations, hurst, seed, initial_std=1.0):

"""
    Generate a 1D fractal boundary curve via midpoint displacement.

    n_iterations: number of recursive subdivisions; curve has
                  2^n_iterations + 1 points.
    hurst:        Hurst exponent H, in (0, 1). Lower H -> rougher,
                  more jagged boundary (higher fractal dimension).
                  Fractal dimension of the resulting curve D = 2 - H.
    seed:         RNG seed, for reproducible but distinct realisations.

"""
    rng = np.random.default_rng(seed)
    n_points = 2 ** n_iterations + 1
    y = np.zeros(n_points)
    # Initialise endpoints
    y[0] = rng.normal(0, initial_std)
    y[-1] = rng.normal(0, initial_std)

    step_size = initial_std

    current_n = n_iterations
    segment_length = n_points - 1

    for iteration in range(1, n_iterations + 1):
        half = segment_length // 2
        displacement_std = step_size * (0.5 ** (iteration * hurst))
        for start in range(0, n_points - 1, segment_length):
            mid = start + half
            end = start + segment_length
            avg = (y[start] + y[end]) / 2.0
            y[mid] = avg + rng.normal(0, displacement_std)
        segment_length = half
```

```
    x = np.linspace(0, 1, n_points)
    return x, y

# ----------------------------------------------------------------------
# 2. Box-counting dimension estimator
# ----------------------------------------------------------------------

def box_counting_dimension(x, y, n_scales=8, min_boxes_per_axis=4):

"""
    Estimate the fractal (box-counting) dimension of a curve (x, y).

Overlays grids of decreasing box size over the curve, counts how many boxes contain at least
one point of the curve, and fits the slope of log(N) vs log(1/epsilon).

    Returns: (dimension_estimate, epsilons, counts, fit_coeffs)
"""
    x_range = x.max() - x.min()
    y_range = y.max() - y.min()
    span = max(x_range, y_range)
    if span == 0:
span = 1.0

# ----------------------------------------------------------------------
# Normalize curve into unit square for consistent box counting
# ----------------------------------------------------------------------

    x_norm = (x - x.min()) / span
    y_norm = (y - y.min()) / span

    max_boxes_per_axis = 2 ** (n_scales + 1)
    box_counts_per_axis = np.geomspace(min_boxes_per_axis, max_boxes_per_axis,
n_scales).astype(int)
    box_counts_per_axis = np.unique(box_counts_per_axis)

    epsilons = []
    counts = []

    for n_boxes in box_counts_per_axis:
        epsilon = 1.0 / n_boxes
        box_x = np.floor(x_norm / epsilon).astype(int)
        box_y = np.floor(y_norm / epsilon).astype(int)
        occupied = set(zip(box_x, box_y))
        epsilons.append(epsilon)
```

```
        counts.append(len(occupied))

    log_inv_eps = np.log(1.0 / np.array(epsilons))
    log_counts = np.log(np.array(counts))

    fit_coeffs = np.polyfit(log_inv_eps, log_counts, 1)
    dimension_estimate = fit_coeffs[0]

    return dimension_estimate, np.array(epsilons), np.array(counts), fit_coeffs

# ---------------------------------------------------------------------
# 3. Generate several distinct realisations from the SAME mechanism
#    ("myriad shapes" from one generative process)
# ---------------------------------------------------------------------

n_iterations_display = 14

# resolution for the four "myriad shapes" panels (visual only)

n_iterations = 17

# resolution for the scale-invariance measurement (needs finer sampling)

hurst = 0.65  # roughness parameter; D_expected = 2 - H
seeds = [1, 2, 3, 4]

realisations = []
for s in seeds:
    x, y = midpoint_displacement(n_iterations_display, hurst, seed=s)
realisations.append((x, y))

# ---------------------------------------------------------------------
# 4. Scale-invariance test: compare fractal dimension of the FULL curve against magnified #
SUB-SECTIONS of the same curve, at several locations, averaged for a robust estimate
# ---------------------------------------------------------------------

x_full, y_full = midpoint_displacement(n_iterations, hurst, seed=100)

dim_full, eps_full, counts_full, fit_full = box_counting_dimension(x_full, y_full, n_scales=10)

# Zoom into several 1/8th-of-domain sub-sections and average the estimate

n_points_full = len(x_full)
zoom_len = n_points_full // 8
```

```
zoom_start_fracs = [1/8, 3/8, 5/8]

zoom_dims = []
zoom_data_for_plot = None
for frac in zoom_start_fracs:
    zoom_start = int(n_points_full * frac)
    x_z = x_full[zoom_start:zoom_start + zoom_len]
    y_z = y_full[zoom_start:zoom_start + zoom_len]
    d, eps_z, counts_z, fit_z = box_counting_dimension(x_z, y_z, n_scales=9)
    zoom_dims.append(d)
    if zoom_data_for_plot is None:  # keep the first window's data for the illustrative plot
        zoom_data_for_plot = (x_z, y_z, eps_z, counts_z, fit_z, zoom_start)
dim_zoom_avg = float(np.mean(zoom_dims))
x_zoom, y_zoom, eps_zoom, counts_zoom, fit_zoom, zoom_start = zoom_data_for_plot
dim_zoom = zoom_dims[0]  # dimension of the specific window shown in the figure

print(f"Expected fractal dimension (D = 2 - H, H = {hurst}): {2 - hurst:.4f}")
print(f"Measured dimension, full curve:              {dim_full:.4f}")
print(f"Measured dimension, individual zoom windows: {[round(d, 4) for d in zoom_dims]}")
print(f"Measured dimension, zoom windows averaged:   {dim_zoom_avg:.4f}")
print(f"Difference (full vs. averaged zoom):         {abs(dim_full - dim_zoom_avg):.4f}")

# ---------------------------------------------------------------------
# 5. Plotting
# ---------------------------------------------------------------------

fig = plt.figure(figsize=(13, 9))
gs = fig.add_gridspec(3, 4, height_ratios=[1, 1, 1.1], hspace=0.55, wspace=0.35)

# Row 1: four distinct realisations, same mechanism ("myriad shapes")

colors = ["#2b5876", "#6a3d9a", "#a33", "#2e8b57"]
for i, (x, y) in enumerate(realisations):
    ax = fig.add_subplot(gs[0, i])
    ax.plot(x, y, color=colors[i], linewidth=1.1)
    ax.fill_between(x, y, y.min(), color=colors[i], alpha=0.12)
    ax.set_title(f"Realisation {i+1} (seed={seeds[i]})", fontsize=9)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.spines[["top", "right", "left", "bottom"]].set_visible(False)

# Row 2: left two panels: full curve with zoom region highlighted

ax_full = fig.add_subplot(gs[1, 0:2])
```

```
ax_full.plot(x_full, y_full, color="#2b5876", linewidth=0.6)
ax_full.axvspan(x_full[zoom_start], x_full[zoom_start + zoom_len - 1], color="#a33",
alpha=0.2)
ax_full.set_title("Full boundary curve (zoom region shaded)", fontsize=9.5)
ax_full.set_xticks([])
ax_full.set_yticks([])
ax_full.spines[["top", "right", "left", "bottom"]].set_visible(False)

ax_zoom = fig.add_subplot(gs[1, 2:4])
ax_zoom.plot(x_zoom, y_zoom, color="#a33", linewidth=1.2)
ax_zoom.set_title("Magnified sub-section (1/16 of domain)", fontsize=9.5)
ax_zoom.set_xticks([])
ax_zoom.set_yticks([])
ax_zoom.spines[["top", "right", "left", "bottom"]].set_visible(False)

# Row 3: box-counting log-log plots for full curve and zoomed section

ax_log_full = fig.add_subplot(gs[2, 0:2])
log_inv_eps_full = np.log(1.0 / eps_full)
log_counts_full = np.log(counts_full)
ax_log_full.scatter(log_inv_eps_full, log_counts_full, color="#2b5876", s=28)
fit_line_full = np.polyval(fit_full, log_inv_eps_full)
ax_log_full.plot(log_inv_eps_full, fit_line_full, color="#2b5876", linestyle="--",
        label=f"slope (D) = {dim_full:.3f}")
ax_log_full.set_xlabel("log(1 / box size)")
ax_log_full.set_ylabel("log(box count)")
ax_log_full.set_title("Box-counting: full curve", fontsize=9.5)
ax_log_full.legend(frameon=False, fontsize=8.5)
ax_log_full.spines[["top", "right"]].set_visible(False)

ax_log_zoom = fig.add_subplot(gs[2, 2:4])
log_inv_eps_zoom = np.log(1.0 / eps_zoom)
log_counts_zoom = np.log(counts_zoom)
ax_log_zoom.scatter(log_inv_eps_zoom, log_counts_zoom, color="#a33", s=28)
fit_line_zoom = np.polyval(fit_zoom, log_inv_eps_zoom)
ax_log_zoom.plot(log_inv_eps_zoom, fit_line_zoom, color="#a33", linestyle="--",
        label=f"slope (D) = {dim_zoom:.3f}\n(avg. across 3 windows =
{dim_zoom_avg:.3f})")
ax_log_zoom.set_xlabel("log(1 / box size)")
ax_log_zoom.set_ylabel("log(box count)")
ax_log_zoom.set_title("Box-counting: zoomed sub-section (1/8 of domain)", fontsize=9.5)
ax_log_zoom.legend(frameon=False, fontsize=8.5)
ax_log_zoom.spines[["top", "right"]].set_visible(False)
```

```
fig.suptitle(
   "Simulation Five: Fractal Boundary Formation - Myriad Shapes from a Single Generative
Mechanism",
   fontsize=12.5, y=0.98
)
fig.savefig("/home/claude/simulation_five/simulation_five_plot.png", dpi=200,
bbox_inches="tight")
print("\nPlot saved.")
```

C.6 - CMB Comparison and Related Robustness Tests (B.6)

cmb_comparison.py

```
"""
cmb_comparison.py

Follow-up test on the "Null Point -> Universe" five-stage emergence pipeline: does the Q-
Brane / Universe-stage output structurally resemble the real Cosmic Microwave Background
(CMB) angular power spectrum, or is the visual similarity just the generic look of any smoothed
correlated-noise field?

Method
------
1. Regenerate the same Q-Brane and Universe-stage fields as the original emergence_pipeline.py
(identical construction, same RNG seed).

2. Compute the *radially-averaged 2D power spectrum* P(k) of each field -- the flat-space
analogue of the CMB's angular power spectrum C_l.

3. Load the real Planck 2018 TT power spectrum (public data release, COM_PowerSpect_CMB-
TT-binned_R3.01.txt, Planck Legacy Archive via IRSA), which is D_l = l(l+1)C_l/2pi vs
multipole l.

4. Compare shapes on equal footing: normalise both curves (x-axis to [0,1] fraction of range, y-
axis to peak=1), then look for the defining structural feature of the real spectrum -- multiple
harmonically-spaced acoustic peaks -- versus a single broad hump.

5. Quantify with peak-counting and a cross-correlation of the log-log slopes, rather than
eyeballing images.

"""
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
from scipy.signal import find_peaks
```

```
rng = np.random.default_rng(7)

L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)

# ----------------------------------------------------------------------
# Regenerate Stage 1 (needed as the basis for Stage 3's k0 scale)
# ----------------------------------------------------------------------

k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])

# ----------------------------------------------------------------------
# Regenerate Stage 3 -- Q-Brane (identical construction/seed to original)
# ----------------------------------------------------------------------

n_waves = 16
qbrane_field = np.zeros((L, L))
dirs = rng.uniform(0, 2 * np.pi, n_waves)
phases_q = rng.uniform(0, 2 * np.pi, n_waves)
for th, ph in zip(dirs, phases_q):
    qbrane_field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
qbrane_field /= np.sqrt(n_waves)

thresholds = np.linspace(qbrane_field.min(), qbrane_field.max(), 80)
giant_frac = []
for t in thresholds:
    mask = qbrane_field > t
    lbl, n = ndimage.label(mask)
    if n == 0:
        giant_frac.append(0.0)
        continue
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
    giant_frac.append(sizes.max() / mask.size)
giant_frac = np.array(giant_frac)

above = np.where(giant_frac > 0.20)[0]
ignite_idx = above[0] if len(above) else int(np.argmax(giant_frac))
t_crit = thresholds[ignite_idx]

mask_crit = qbrane_field > t_crit
lbl_crit, n_crit = ndimage.label(mask_crit)
```

```
sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
giant_label = int(np.argmax(sizes_crit)) + 1
giant_mask = lbl_crit == giant_label

# ----------------------------------------------------------------------
# Regenerate Stage 4 -- Universe
# ----------------------------------------------------------------------

universe_mask = ndimage.binary_closing(giant_mask, iterations=2)
universe_field = ndimage.gaussian_filter(universe_mask.astype(float), sigma=1.2)

print("Regenerated Q-Brane and Universe-stage fields (matches original pipeline).")

# ----------------------------------------------------------------------
# Radially-averaged 2D power spectra (flat-space analogue of C_l)
# ----------------------------------------------------------------------

def radial_power_spectrum(field, n_bins=60):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    ny, nx = P.shape
    cy, cx = ny // 2, nx // 2
    yy, xx = np.indices((ny, nx))
    r = np.sqrt((yy - cy) ** 2 + (xx - cx) ** 2)
    r_max = min(cy, cx)
    bins = np.linspace(0, r_max, n_bins + 1)
    bin_idx = np.digitize(r.ravel(), bins)
    radial_P = np.array([
        P.ravel()[bin_idx == i].mean() if np.any(bin_idx == i) else 0.0
        for i in range(1, n_bins + 1)
    ])
    k = 0.5 * (bins[1:] + bins[:-1])
    return k, radial_P

k_q, P_q = radial_power_spectrum(qbrane_field)
k_u, P_u = radial_power_spectrum(universe_field)

# Drop the k=0 (DC) bin, which is dominated by the mean value and carries no structural
# information -- same treatment the original pipeline used.

k_q, P_q = k_q[1:], P_q[1:]
k_u, P_u = k_u[1:], P_u[1:]
```

```
# ----------------------------------------------------------------------
# Load the real Planck 2018 TT power spectrum
# ----------------------------------------------------------------------

planck = np.loadtxt("/home/claude/cmb_compare/planck_tt_binned.txt")
ell, Dl = planck[:, 0], planck[:, 1]

# ----------------------------------------------------------------------
# Put all three curves on equal footing: x normalised to [0,1] over its own range, y normalised
# so the peak = 1. This is the only fair way to compare a k-space spectrum on a 300x300 flat grid
# against a multipole spectrum running from l=2 to l=2500 -- absolute scales are meaningless
# across the two systems, but *shape* (number and sharpness of peaks, rise/fall pattern) is
# exactly what a visual "this looks like the CMB" claim is actually asserting.
# ----------------------------------------------------------------------

def normalise(xarr, yarr):
    xn = (xarr - xarr.min()) / (xarr.max() - xarr.min())
    yn = yarr / yarr.max()
    return xn, yn

xn_planck, yn_planck = normalise(ell, Dl)
xn_q, yn_q = normalise(k_q, P_q)
xn_u, yn_u = normalise(k_u, P_u)

# ----------------------------------------------------------------------
# Peak counting -- the defining structural signature of the real CMB spectrum is a series of
# harmonically-spaced acoustic peaks (a fundamental + overtones from photon-baryon
# oscillations). Count prominent peaks in each normalised curve on equal terms.
# ----------------------------------------------------------------------

def count_peaks(y, prominence=0.08):
    peaks, props = find_peaks(y, prominence=prominence)
    return len(peaks), peaks

n_peaks_planck, peaks_planck = count_peaks(yn_planck)
n_peaks_q, peaks_q = count_peaks(yn_q)
n_peaks_u, peaks_u = count_peaks(yn_u)

print(f"\nPeak count (prominence >= 0.08 of normalised max):")
print(f"  Planck 2018 TT spectrum : {n_peaks_planck} acoustic peaks")
print(f"  Q-Brane field spectrum  : {n_peaks_q} peak(s)")
print(f"  Universe field spectrum : {n_peaks_u} peak(s)")
```

```
# ----------------------------------------------------------------------
# Log-log slope comparison in the falling (high-k / high-l) regime, since that's the region
# both a percolation cluster and a diffusion-damped CMB tail will show a power-law-ish
# falloff.
# ----------------------------------------------------------------------

def falling_slope(xarr, yarr, frac_start=0.3):

# ----------------------------------------------------------------------
# Only look at the declining part beyond the first peak.
# ----------------------------------------------------------------------

    peak_i = np.argmax(yarr)
    xs, ys = xarr[peak_i:], yarr[peak_i:]
    valid = ys > 0
    xs, ys = xs[valid], ys[valid]
    if len(xs) < 4:
        return np.nan
    logx, logy = np.log10(xs[1:] + 1e-9), np.log10(ys[1:] + 1e-9)
    slope, _ = np.polyfit(logx, logy, 1)
    return slope

slope_planck = falling_slope(ell, Dl)
slope_q = falling_slope(k_q, P_q)
slope_u = falling_slope(k_u, P_u)

print(f"\nHigh-l / high-k falloff slope (log-log):")
print(f"  Planck 2018 TT spectrum : {slope_planck:.2f}")
print(f"  Q-Brane field spectrum  : {slope_q:.2f}")
print(f"  Universe field spectrum : {slope_u:.2f}")

# ----------------------------------------------------------------------
# Figure: normalised spectra side by side
# ----------------------------------------------------------------------

fig, axes = plt.subplots(1, 2, figsize=(13, 5))

ax0 = axes[0]
ax0.plot(xn_planck, yn_planck, color='#1a1a2e', linewidth=2, label='Planck 2018 TT (real
CMB)')
ax0.plot(xn_q, yn_q, color='#c9963d', linewidth=2, label='Q-Brane field (simulation)')
ax0.plot(xn_u, yn_u, color='#8b3a3a', linewidth=2, label='Universe field (simulation)')
ax0.plot(peaks_planck / (len(xn_planck) - 1), yn_planck[peaks_planck], 'o', color='#1a1a2e',
markersize=6)
```

```
ax0.plot(peaks_q / (len(xn_q) - 1), yn_q[peaks_q], 'o', color='#c9963d', markersize=6)
ax0.plot(peaks_u / (len(xn_u) - 1), yn_u[peaks_u], 'o', color='#8b3a3a', markersize=6)
ax0.set_xlabel("normalised scale (0=largest, 1=smallest)")
ax0.set_ylabel("normalised power (peak = 1)")
ax0.set_title("Power spectra, normalised for shape comparison")
ax0.legend(fontsize=8)
ax0.grid(alpha=0.25)

ax1 = axes[1]
ax1.loglog(ell, Dl / Dl.max(), color='#1a1a2e', linewidth=2, label='Planck 2018 TT')
ax1.loglog(k_q, P_q / P_q.max(), color='#c9963d', linewidth=2, label='Q-Brane field')
ax1.loglog(k_u, P_u / P_u.max(), color='#8b3a3a', linewidth=2, label='Universe field')
ax1.set_xlabel("scale (log)")
ax1.set_ylabel("normalised power (log)")
ax1.set_title("Same data, log-log")
ax1.legend(fontsize=8)
ax1.grid(alpha=0.25, which='both')

fig.suptitle("Does the Q-Brane/Universe simulation match the real CMB spectrum's shape?",
fontsize=12)
fig.tight_layout(rect=[0, 0, 1, 0.95])
fig.savefig("/home/claude/cmb_compare/cmb_comparison.png", dpi=170, facecolor='white')
print("\nSaved figure to cmb_comparison.png")
cmb_comparison_full.py

"""
cmb_comparison_full.py

Runs the real CMB comparison across all five stages of the emergence pipeline (Null Point ->
Quasi-Crystalline Substrate -> Quantum Foam -> Q-Brane -> Universe), not just the last two.

Fields are regenerated with the identical construction and RNG call order as the original
emergence_pipeline.py so the Q-Brane/Universe fields are the exact ones that appear in the plate
images (the RNG is a single shared stream, so skipping the quantum-foam stage's draws -- as
the first, quick comparison did -- silently changes every draw after it). This version consumes the
RNG in the original order: substrate (no draws) -> foam (many draws) -> q-brane (16+16
draws).

"""
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
from scipy.signal import find_peaks
```

```
rng = np.random.default_rng(7)

L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)

# ----------------------------------------------------------------------
# Stage 0 -- Null Point: no field, no structure. Coherence 0 by construction. There is
# nothing to Fourier-transform, so it is excluded from the spectral comparison and just
# noted as a baseline.
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Stage 1 -- Quasi-Crystalline Substrate (deterministic, no RNG draws)
# ----------------------------------------------------------------------

k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])

substrate_field = np.zeros((L, L))
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))

# ----------------------------------------------------------------------
# Stage 2 -- Quantum Foam (consumes RNG draws, same order as original)
# ----------------------------------------------------------------------

n_real = 40
foam_snapshot = None
for i in range(n_real):
    phases = rng.uniform(0, 2 * np.pi, size=5)
    amps = 1 + 0.35 * rng.normal(size=5)
    field = np.zeros((L, L))
    for th, ph, a in zip(thetas, phases, amps):
        field += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    field += 0.9 * rng.normal(size=(L, L))
    if i == n_real - 1:
        foam_snapshot = field

# ----------------------------------------------------------------------
# Stage 3 -- Q-Brane (RNG draws continue on from Stage 2, same order)
# ----------------------------------------------------------------------

n_waves = 16
```

```
qbrane_field = np.zeros((L, L))
dirs = rng.uniform(0, 2 * np.pi, n_waves)
phases_q = rng.uniform(0, 2 * np.pi, n_waves)
for th, ph in zip(dirs, phases_q):
    qbrane_field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
qbrane_field /= np.sqrt(n_waves)
thresholds = np.linspace(qbrane_field.min(), qbrane_field.max(), 80)
giant_frac = []
for t in thresholds:
    mask = qbrane_field > t
    lbl, n = ndimage.label(mask)
    if n == 0:
        giant_frac.append(0.0)
        continue
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
    giant_frac.append(sizes.max() / mask.size)
giant_frac = np.array(giant_frac)

above = np.where(giant_frac > 0.20)[0]
ignite_idx = above[0] if len(above) else int(np.argmax(giant_frac))
t_crit = thresholds[ignite_idx]

mask_crit = qbrane_field > t_crit
lbl_crit, n_crit = ndimage.label(mask_crit)
sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
giant_label = int(np.argmax(sizes_crit)) + 1
giant_mask = lbl_crit == giant_label

# ---------------------------------------------------------------------
# Stage 4 -- Universe
# ---------------------------------------------------------------------

universe_mask = ndimage.binary_closing(giant_mask, iterations=2)
universe_field = ndimage.gaussian_filter(universe_mask.astype(float), sigma=1.2)

print("Regenerated all four structured stages (RNG stream now matches the original pipeline).")

# ---------------------------------------------------------------------
# Radially-averaged 2D power spectrum
# ---------------------------------------------------------------------

def radial_power_spectrum(field, n_bins=60):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
```

```
    ny, nx = P.shape
    cy, cx = ny // 2, nx // 2
    yy, xx = np.indices((ny, nx))
    r = np.sqrt((yy - cy) ** 2 + (xx - cx) ** 2)
    r_max = min(cy, cx)
    bins = np.linspace(0, r_max, n_bins + 1)
    bin_idx = np.digitize(r.ravel(), bins)
    radial_P = np.array([
        P.ravel()[bin_idx == i].mean() if np.any(bin_idx == i) else 0.0
        for i in range(1, n_bins + 1)
    ])
    k = 0.5 * (bins[1:] + bins[:-1])
    return k[1:], radial_P[1:]  # drop DC bin

fields = {
    "Quasi-Crystalline Substrate": substrate_field,
    "Quantum Foam": foam_snapshot,
    "Q-Brane": qbrane_field,
    "Universe": universe_field,
}

spectra = {name: radial_power_spectrum(f) for name, f in fields.items()}

# ---------------------------------------------------------------------
# Real Planck 2018 TT power spectrum
# ---------------------------------------------------------------------

planck = np.loadtxt("/home/claude/cmb_compare/planck_tt_binned.txt")
ell, Dl = planck[:, 0], planck[:, 1]

def normalise(xarr, yarr):
    xn = (xarr - xarr.min()) / (xarr.max() - xarr.min())
    yn = yarr / yarr.max()
    return xn, yn

def count_peaks(y, prominence=0.08):
    peaks, _ = find_peaks(y, prominence=prominence)
    return len(peaks), peaks

def falling_slope(xarr, yarr):
    peak_i = np.argmax(yarr)
    xs, ys = xarr[peak_i:], yarr[peak_i:]
    valid = ys > 0
    xs, ys = xs[valid], ys[valid]
```

```
    if len(xs) < 4:
        return np.nan
    logx, logy = np.log10(xs[1:] + 1e-9), np.log10(ys[1:] + 1e-9)
    slope, _ = np.polyfit(logx, logy, 1)
    return slope
xn_planck, yn_planck = normalise(ell, Dl)
n_peaks_planck, peaks_planck = count_peaks(yn_planck)
slope_planck = falling_slope(ell, Dl)

print(f"\n{'Stage':30s} {'peaks':>6s} {'falloff slope':>15s}")
print(f"{'Planck 2018 TT (real CMB)':30s} {n_peaks_planck:6d} {slope_planck:15.2f}")

results = {}
for name, (k, P) in spectra.items():
    xn, yn = normalise(k, P)
    npk, pk = count_peaks(yn)
    slope = falling_slope(k, P)
    results[name] = (xn, yn, pk, npk, slope)
    print(f"{name:30s} {npk:6d} {slope:15.2f}")

# ----------------------------------------------------------------------
# Figure: five-panel comparison
# ----------------------------------------------------------------------

fig, axes = plt.subplots(1, 5, figsize=(22, 4.2), sharey=True)

ax0 = axes[0]
ax0.plot(xn_planck, yn_planck, color='#1a1a2e', linewidth=2)
ax0.plot(peaks_planck / (len(xn_planck) - 1), yn_planck[peaks_planck], 'o', color='#1a1a2e')
ax0.set_title(f"Planck 2018 TT\n(real CMB) - {n_peaks_planck} peaks", fontsize=10)
ax0.set_ylabel("normalised power")
ax0.grid(alpha=0.25)

colors = ['#4a7a8c', '#8b3a3a', '#c9963d', '#5a8c4a']
for ax, (name, (xn, yn, pk, npk, slope)) in zip(axes[1:], Results:items()):
    ax.plot(xn, yn, color=colors[list(Results:keys()).index(name)], linewidth=2)
    ax.plot(pk / (len(xn) - 1), yn[pk], 'o', color=colors[list(Results:keys()).index(name)])
    ax.set_title(f"{name}\n{npk} peak(s), slope {slope:.1f}", fontsize=10)
    ax.set_xlabel("normalised scale")
    ax.grid(alpha=0.25)

fig.suptitle("Real CMB spectrum vs. all four structured pipeline stages", fontsize=13)
fig.tight_layout(rect=[0, 0, 1, 0.92])
```

```
fig.savefig("/home/claude/cmb_compare/cmb_comparison_full.png", dpi=170,
facecolor='white')
print("\nSaved figure to cmb_comparison_full.png")
theory_tests.py

"""
theory_tests.py
Two rigorous tests against established, quantitative theoretical predictions -- as opposed to
comparing images by eye.

TEST 1: Does the Q-Brane percolation transition belong to the standard 2D percolation
universality class? Known exact values (site percolation, 2D, infinite system):
      tau   (cluster-size distribution exponent) = 187/91 = 2.0549...
      D     (fractal dimension of spanning cluster) = 91/48 = 1.8958...
      gamma (susceptibility exponent) = 43/18 = 2.3889...
These are measured here from an ensemble of independently-generated Q-Brane-style fields
(same construction as the pipeline, many random-phase realizations), not just the single field
used for the manuscript plate -- exponents are only meaningful as ensemble/scaling properties,
not properties of one sample.

TEST 2: Is any pipeline field consistent with a near-scale-invariant primordial power spectrum,
      P(k) ~ k^(n_s - 1), n_s = 0.9649 +/- 0.0042
      (Planck 2018 best-fit)?
This is a much simpler, single-number theoretical target than the full acoustic-peak spectrum,
and does not require any baryon-photon oscillator physics -- just: is the field's power spectrum
close to flat (in this convention) over a decade or more of scales?

"""
import numpy as np
from scipy import ndimage

rng_master = np.random.default_rng(42)
L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)
k0 = 2.0

# ---------------------------------------------------------------------
# TEST 1 -- Percolation universality class
# ---------------------------------------------------------------------

def make_qbrane_field(seed_rng, n_waves=16):
    field = np.zeros((L, L))
    dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
```

```
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

N_REALIZATIONS = 25
all_cluster_sizes_at_criticality = []
giant_masks_at_criticality = []

# ---------------------------------------------------------------------
# susceptibility (mean size of non-giant clusters) vs threshold, ensemble avg
# ---------------------------------------------------------------------

n_thresh = 150
susceptibility_accum = np.zeros(n_thresh)
giant_frac_accum = np.zeros(n_thresh)
thresholds_ref = None
for r in range(N_REALIZATIONS):
    field = make_qbrane_field(rng_master)
    thresholds = np.linspace(field.min(), field.max(), n_thresh)
    if thresholds_ref is None:
        thresholds_ref = thresholds

    giant_frac = np.zeros(n_thresh)
    suscept = np.zeros(n_thresh)
    for i, t in enumerate(thresholds):
        mask = field > t
        lbl, n = ndimage.label(mask)
        if n == 0:
            continue
        sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
        giant_frac[i] = sizes.max() / mask.size
        non_giant = np.delete(sizes, np.argmax(sizes))
        suscept[i] = non_giant.mean() if len(non_giant) else 0.0

    giant_frac_accum += giant_frac
susceptibility_accum += suscept

# ---------------------------------------------------------------------
# NOTE: the original pipeline located "ignition" as the first threshold where giant_frac > 0.20.
# Since the field starts fully occupied at the lowest threshold (giant_frac ~= 1.0 there), that rule is
# trivially satisfied at the very first sample -- it never actually finds a transition. The real
# percolation transition is where giant_frac is dropping fastest as threshold increases (the
# standard finite-system proxy for the critical point), so that is used here instead.
```

```
# ----------------------------------------------------------------------

    grad = np.gradient(giant_frac, thresholds)
    ignite_idx = int(np.argmin(grad))
    t_crit = thresholds[ignite_idx]
    mask_crit = field > t_crit
    lbl_crit, n_crit = ndimage.label(mask_crit)
    sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
    giant_label = int(np.argmax(sizes_crit)) + 1
    giant_masks_at_criticality.append(lbl_crit == giant_label)
    non_giant_sizes = np.delete(sizes_crit, np.argmax(sizes_crit))
    all_cluster_sizes_at_criticality.extend(non_giant_sizes[non_giant_sizes >= 1].tolist())

giant_frac_avg = giant_frac_accum / N_REALIZATIONS
susceptibility_avg = susceptibility_accum / N_REALIZATIONS

print(f"Ran {N_REALIZATIONS} independent Q-Brane-field realizations.\n")

# ----------------------------------------------------------------------
# --- tau: cluster-size distribution exponent (MLE for power-law tail) ---
# ----------------------------------------------------------------------

sizes_arr = np.array(all_cluster_sizes_at_criticality, dtype=float)
sizes_arr = sizes_arr[sizes_arr >= 2]  # drop size-1 clusters (discretization noise)
xmin = 2.0
n = len(sizes_arr)
tau_hat = 1 + n / np.sum(np.log(sizes_arr / xmin))
tau_stderr = (tau_hat - 1) / np.sqrt(n)

print("--- Cluster-size distribution exponent (tau) ---")
print(f"  measured tau = {tau_hat:.3f} +/- {tau_stderr:.3f}  (n={n} clusters pooled)")
print(f"  2D percolation theory: tau = 187/91 = {187/91:.4f}")
print(f"  deviation: {abs(tau_hat - 187/91):.3f} ({abs(tau_hat - 187/91)/tau_stderr:.1f} sigma)\n")

# ----------------------------------------------------------------------
# --- D: fractal dimension of the spanning/giant cluster (box counting) ---
# ----------------------------------------------------------------------

def box_count_dimension(mask, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(mask.shape) // 4
    sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in sizes:
```

```
        ny, nx = mask.shape[0] // bs, mask.shape[1] // bs
        trimmed = mask[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occupied = blocks.any(axis=(1, 3))
        counts.append(occupied.sum())
    counts = np.array(counts)
    valid = counts > 0
    logs_inv_box = np.log(1.0 / sizes[valid])
    log_counts = np.log(counts[valid])
    slope, _ = np.polyfit(logs_inv_box, log_counts, 1)
    return slope
D_estimates = [box_count_dimension(m) for m in giant_masks_at_criticality]
D_mean, D_std = np.mean(D_estimates), np.std(D_estimates) / np.sqrt(len(D_estimates))

print("--- Fractal dimension of spanning cluster (D) ---")
print(f"  measured D = {D_mean:.3f} +/- {D_std:.3f}  (box-counting, {len(D_estimates)}
realizations)")
print(f"  2D percolation theory: D = 91/48 = {91/48:.4f}")
print(f"  deviation: {abs(D_mean - 91/48):.3f} ({abs(D_mean - 91/48)/D_std:.1f} sigma)\n")

# ---------------------------------------------------------------------
# --- gamma: susceptibility exponent, fit log(susceptibility) vs log|t-t_c| ---
# ---------------------------------------------------------------------

t_c_global = thresholds_ref[np.argmax(np.gradient(giant_frac_avg))]
delta = np.abs(thresholds_ref - t_c_global)
valid = (delta > 1e-6) & (susceptibility_avg > 0)
logd = np.log(delta[valid])
logs = np.log(susceptibility_avg[valid])
gamma_hat, intercept = np.polyfit(logd, -logs, 1)  # susceptibility ~ delta^-gamma

print("--- Susceptibility exponent (gamma) ---")
print(f"  measured gamma =~ {gamma_hat:.3f}  (rough single-run fit, not finite-size-scaled)")
print(f"  2D percolation theory: gamma = 43/18 = {43/18:.4f}\n")

# ---------------------------------------------------------------------
# TEST 2 -- Scale invariance vs primordial spectral index n_s
# ---------------------------------------------------------------------

print("=" * 60)
print("Scale-invariance check against Planck n_s = 0.9649 +/- 0.0042")
print("=" * 60)

def radial_power_spectrum(field, n_bins=60):
```

```
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    ny, nx = P.shape
    cy, cx = ny // 2, nx // 2
    yy, xx = np.indices((ny, nx))
    r = np.sqrt((yy - cy) ** 2 + (xx - cx) ** 2)
    r_max = min(cy, cx)
    bins = np.linspace(0, r_max, n_bins + 1)
    bin_idx = np.digitize(r.ravel(), bins)
    radial_P = np.array([
        P.ravel()[bin_idx == i].mean() if np.any(bin_idx == i) else 0.0
        for i in range(1, n_bins + 1)
    ])
    k = 0.5 * (bins[1:] + bins[:-1])
    return k[1:], radial_P[1:]

substrate_field = np.zeros((L, L))
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))

rng_demo = np.random.default_rng(7)
n_real = 40
foam_snapshot = None
for i in range(n_real):
    phases = rng_demo.uniform(0, 2 * np.pi, size=5)
    amps = 1 + 0.35 * rng_demo.normal(size=5)
    field = np.zeros((L, L))
    for th, ph, a in zip(thetas, phases, amps):
        field += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    field += 0.9 * rng_demo.normal(size=(L, L))
    if i == n_real - 1:
        foam_snapshot = field
qbrane_demo = make_qbrane_field(rng_demo)

n_s = 0.9649
target_slope = n_s - 1  # ~ -0.0351

for name, f in [("Quasi-Crystalline Substrate", substrate_field),
                ("Quantum Foam", foam_snapshot),
                ("Q-Brane", qbrane_demo)]:
    k, P = radial_power_spectrum(f)
```

```
# ------------------------------------------------------------------
# fit over the mid-range of scales, away from the single dominant peak, the region a "spectral
# index" fit would actually be applied to
# ------------------------------------------------------------------

    peak_i = np.argmax(P)
    lo = min(peak_i + 3, len(k) - 8)
    hi = len(k) - 2
    if hi - lo < 4:
        print(f"{name}: not enough range past the peak to fit a slope")
        continue
    logk, logP = np.log10(k[lo:hi]), np.log10(P[lo:hi] + 1e-12)
    slope, _ = np.polyfit(logk, logP, 1)
    print(f"{name:30s} fitted local slope = {slope:+.3f}   "
          f"(target for scale-invariance: {target_slope:+.3f})")

print("\nDone.")
dla_test.py

"""
dla_test.py

Diffusion-limited aggregation (DLA): the classic single-seed fractal growth model. A single
occupied seed sits at the origin; particles are released one at a time from a launch circle and take
a random walk until they touch the growing cluster, where they stick permanently.

Repeated many times, this grows a fractal cluster from one point -- literally "starting at zero,
proceeding by fractal growth."

Classic result (Witten & Sander 1981): 2D DLA clusters have a fractal dimension of
approximately 1.71.

This measures the fractal dimension of a real DLA cluster with the EXACT SAME box-counting
method used on the Q-Brane/percolation giant clusters all day, so the comparison is apples to
apples.

"""
import numpy as np
from scipy import ndimage

rng = np.random.default_rng(99)
GRID = 401
CENTER = GRID // 2
N_PARTICLES = 4000
```

```
KILL_MARGIN = 30  # particles walking this far past the launch radius are discarded and
relaunched

occupied = np.zeros((GRID, GRID), dtype=bool)
occupied[CENTER, CENTER] = True
max_radius = 1.0

# ---------------------------------------------------------------------
# 8-connected neighborhood for "did I touch the cluster" checks
# ---------------------------------------------------------------------

neighbor_offsets = [(-1, -1), (-1, 0), (-1, 1),
                    (0, -1),           (0, 1),
                    (1, -1),  (1, 0),  (1, 1)]

def touches_cluster(y, x):
    for dy, dx in neighbor_offsets:
        yy, xx = y + dy, x + dx
        if 0 <= yy < GRID and 0 <= xx < GRID and occupied[yy, xx]:
            return True
    return False

stuck_count = 0
attempts = 0
max_attempts = N_PARTICLES * 40

while stuck_count < N_PARTICLES and attempts < max_attempts:
    attempts += 1
    launch_r = max_radius + 8
    kill_r = launch_r + KILL_MARGIN

    if launch_r + KILL_MARGIN >= GRID / 2 - 2:
        # cluster has grown to fill the available grid; stop early
        break

    theta = rng.uniform(0, 2 * np.pi)
    y = CENTER + launch_r * np.sin(theta)
    x = CENTER + launch_r * np.cos(theta)
    while True:
        dist_to_center = np.sqrt((y - CENTER) ** 2 + (x - CENTER) ** 2)
        if dist_to_center > kill_r:
break  # wandered too far; discard and relaunch
```

```
# ---------------------------------------------------------------------
# adaptive step size: take bigger strides when far from the cluster (standard DLA speed
# optimization), small steps once close
# ---------------------------------------------------------------------

        dist_to_cluster_surface = max(dist_to_center - max_radius, 1.0)
        step = max(1, min(int(dist_to_cluster_surface / 2), 8))

        angle = rng.uniform(0, 2 * np.pi)
        y_new = y + step * np.sin(angle)
        x_new = x + step * np.cos(angle)

        yi, xi = int(round(y_new)), int(round(x_new))
        if not (0 <= yi < GRID and 0 <= xi < GRID):
            break

        if touches_cluster(yi, xi):
            occupied[yi, xi] = True
            stuck_count += 1
            r = np.sqrt((yi - CENTER) ** 2 + (xi - CENTER) ** 2)
            if r > max_radius:
                max_radius = r
            break
        y, x = y_new, x_new

print(f"DLA cluster grown: {stuck_count} particles stuck ({attempts} attempts), "
f"max radius = {max_radius:.1f} px")

# ---------------------------------------------------------------------
# Fractal dimension -- identical box-counting method used all day
# ---------------------------------------------------------------------

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
```

```
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope

D_dla = box_count_dimension(occupied)

print(f"\nDLA fractal dimension (measured)      = {D_dla:.3f}")
print(f"DLA fractal dimension (Witten-Sander)  = ~1.71 (classic literature value)")
print(f"\nFor comparison, today's Q-Brane giant-cluster D measurements ranged")
print(f"~1.73 to ~1.79 across the wave-count, domain-size, resolution, and")
print(f"symmetry sweeps.")

# ---------------------------------------------------------------------
# Save an image for visual comparison
# ---------------------------------------------------------------------

import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(6, 6))
ax.imshow(occupied, cmap='inferno', origin='lower')
ax.set_title(f"DLA cluster, {stuck_count} particles, D={D_dla:.3f}")
ax.set_xticks([]); ax.set_yticks([])
fig.tight_layout()
fig.savefig("/home/claude/cmb_compare/dla_cluster.png", dpi=150, facecolor='white')
print("\nSaved cluster image to dla_cluster.png")
fibonacci_spiral_test.py

"""
fibonacci_spiral_test.py

Tests a genuinely different generative principle from either fractal random-walk growth (DLA)
or random-wave percolation (Q-Brane): the Fibonacci/Vogel spiral -- the golden-angle point-
placement rule behind sunflower seed heads, pinecones, and phyllotaxis generally.

Construction (Vogel, 1979):
theta_n = n * golden_angle
golden_angle = 2*pi*(1 - 1/phi)
r_n     = sqrt(n)

This grows deterministically outward from a single point (n=0) -- literally "starting at zero,
proceeding by a single additive rule."

Two things are tested, both using the exact same measurement methods used earlier today:
```

1. Rotational/diffraction symmetry -- does the point pattern's Fourier structure factor show the same kind of sharp, symmetric peaks the quasicrystalline Substrate's 5-plane-wave construction showed (order without simple periodicity)? Golden-angle spirals are known in optics to produce a distinctive pentagonal/decagonal diffraction pattern -- this checks whether that shows up quantitatively here.

2. Fractal dimension via box-counting, for comparison against:
- Q-Brane / percolation giant clusters: D ~ 1.73-1.79 (today)
- DLA fractal growth: D ~ 1.71 (textbook), 1.45 (measured)
- Standard 2D percolation theory: D = 1.896

"""

```
import numpy as np
from scipy import ndimage

phi = (1 + np.sqrt(5)) / 2
golden_angle = 2 * np.pi * (1 - 1 / phi)

N_POINTS = 4000

n = np.arange(N_POINTS)
theta = n * golden_angle
r = np.sqrt(n)
xs = r * np.cos(theta)
ys = r * np.sin(theta)

print(f"Golden angle = {np.degrees(golden_angle):.4f} degrees")
print(f"Grew a {N_POINTS}-point Fibonacci/Vogel spiral from the origin.\n")

# ----------------------------------------------------------------------
# Rasterize onto a grid (needed for both the structure factor and the box-counting
# dimension)
# ----------------------------------------------------------------------

GRID = 400
extent = r.max() * 1.05
xi = ((xs + extent) / (2 * extent) * (GRID - 1)).astype(int)
yi = ((ys + extent) / (2 * extent) * (GRID - 1)).astype(int)
raster = np.zeros((GRID, GRID), dtype=bool)
raster[yi, xi] = True
```

```
# ------------------------------------------------------------------
# TEST 1 -- rotational/diffraction symmetry, same style of test used on the quasicrystalline
# Substrate (structure factor + rotational autocorrelation, not the earlier buggy peak-
# angle-folding method)
# ------------------------------------------------------------------

F = np.fft.fftshift(np.fft.fft2(raster.astype(float)))
P = np.abs(F) ** 2
cy, cx = np.array(P.shape) // 2
P[cy - 2:cy + 3, cx - 2:cx + 3] = 0  # zero out DC peak

# ------------------------------------------------------------------
# Full-image rotational autocorrelation: correlate the power spectrum with rotated copies of
# itself; periodicity in the correlation vs. rotation angle reveals the rotational symmetry order.
# ------------------------------------------------------------------

angles_to_test = np.arange(0, 181, 1)
correlations = []
P_norm = (P - P.mean()) / (P.std() + 1e-12)
for ang in angles_to_test:
    rotated = ndimage.rotate(P_norm, ang, reshape=False, order=1, mode='constant')
    corr = np.mean(P_norm * rotated)
    correlations.append(corr)
correlations = np.array(correlations)

# ------------------------------------------------------------------
# Find the rotational order: look for the smallest angle (other than 0) giving a strong correlation
# peak, and see how many peaks fit in 360.
# ------------------------------------------------------------------

from scipy.signal import find_peaks
peaks, props = find_peaks(correlations[1:], prominence=0.05 * correlations[0])
peaks = peaks + 1  # offset since we sliced [1:]

print("--- Rotational symmetry test (structure factor autocorrelation) ---")
if len(peaks) > 0:
    peak_angles = angles_to_test[peaks]
    print(f"  Autocorrelation peaks at angles: {peak_angles[:10]} degrees")
    if len(peak_angles) > 0:
        implied_fold = 360 / peak_angles[0] if peak_angles[0] > 0 else None
        print(f"  First peak at {peak_angles[0]} deg -> implies ~{implied_fold:.1f}-fold symmetry")
else:
    print("  No significant rotational autocorrelation peaks found "
"(pattern does not show discrete rotational order at this resolution)")
```

```
# ----------------------------------------------------------------------
# TEST 2 -- fractal dimension via box counting (same method all day)
# ----------------------------------------------------------------------

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope

D_fib = box_count_dimension(raster)

print(f"\n--- Fractal dimension (box counting) ---")
print(f"  Fibonacci spiral (measured)       = {D_fib:.3f}")
print(f"  (a fully space-filling 2D pattern approaches D=2; a value")
print(f"   near 2 here means the spiral densely tiles the plane rather")
print(f"   than forming a sparse fractal structure)")
print(f"\n  For comparison:")
print(f"    Q-Brane / percolation clusters (today) : D ~ 1.73-1.79")
print(f"    DLA fractal growth (measured today)    : D = 1.450")
print(f"    DLA fractal growth (textbook)          : D ~ 1.71")
print(f"    Standard 2D percolation (theory)       : D = 1.896")
# Save an image

import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 2, figsize=(12, 5.5))
ax[0].scatter(xs, ys, s=1, color='#c9963d')
ax[0].set_aspect('equal')
ax[0].set_title(f"Fibonacci/Vogel spiral\n({N_POINTS} points, golden angle)")
ax[0].set_xticks([]); ax[0].set_yticks([])

ax[1].plot(angles_to_test, correlations, color='#1a1a2e')
```

```
ax[1].set_xlabel("rotation angle (degrees)")
ax[1].set_ylabel("structure-factor autocorrelation")
ax[1].set_title("Rotational symmetry test")
ax[1].grid(alpha=0.3)

fig.tight_layout()
fig.savefig("/home/claude/cmb_compare/fibonacci_spiral.png", dpi=150, facecolor='white')
print("\nSaved figure to fibonacci_spiral.png")
```

pentagon_phi_test.py

```
"""
```

pentagon_phi_test.py

Tests the pentagon/golden-ratio connection directly on the Substrate stage's actual construction (5 plane waves at 72-degree increments) -- not a new spiral, the real thing already in the pipeline.

TEST 1 -- the geometric identity itself.
A regular pentagon's diagonal-to-side ratio is exactly phi. The five unit wavevectors used in the Substrate (at 0, 72, 144, 216, 288 degrees) ARE the vertices of a regular pentagon on the unit circle.
This confirms that identity numerically, directly on the actual vectors used in the construction -- establishing the connection is real and not just a coincidence of two unrelated numbers.

TEST 2 -- the Penrose-tiling golden-ratio count test.
A famous, exact result for Penrose tilings: as a tiling grows, the RATIO of the number of "fat" rhombus tiles to "thin" rhombus tiles approaches phi. This checks whether the Substrate field's nodal domains (the connected regions where the field is positive, versus where it's negative) show an analogous two-population structure whose COUNT RATIO approaches phi as the sampled domain grows -- the same kind of finite-size sweep used on the Q-Brane tests earlier today.

```
"""
import numpy as np
from scipy import ndimage

phi = (1 + np.sqrt(5)) / 2

# ----------------------------------------------------------------------
# TEST 1 -- pentagon geometry IS phi, directly on the Substrate's vectors
# ----------------------------------------------------------------------

print("=" * 65)
print("TEST 1: the Substrate's 5 wavevectors are a regular pentagon")
```

```
print("=" * 65)

thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])
vertices = np.stack([np.cos(thetas), np.sin(thetas)], axis=1)
side = np.linalg.norm(vertices[1] - vertices[0])       # adjacent vertices
diagonal = np.linalg.norm(vertices[2] - vertices[0])   # skip-one vertices
ratio = diagonal / side

print(f"  side length (adjacent vertices)    = {side:.6f}")
print(f"  diagonal length (skip-one vertices) = {diagonal:.6f}")
print(f"  diagonal / side                     = {ratio:.6f}")
print(f"  phi                                 = {phi:.6f}")
print(f"  match: {'YES' if abs(ratio - phi) < 1e-9 else 'NO'} (difference = {abs(ratio-phi):.2e})")

print(f"\n  Also: cos(72 deg) = {np.cos(np.radians(72)):.6f},  (phi-1)/2 = {(phi-1)/2:.6f}")
print(f"        cos(36 deg) = {np.cos(np.radians(36)):.6f},  phi/2     = {phi/2:.6f}")
print("\n  So yes -- the 72-degree spacing used to build the Substrate field")
print("  is not an arbitrary angle. It IS the golden ratio, geometrically.")
print("  This part of the connection is exact, not approximate.")

# ---------------------------------------------------------------------
# TEST 2 -- do nodal-domain populations show the Penrose fat/thin ratio?
# ---------------------------------------------------------------------

print("\n" + "=" * 65)
print("TEST 2: nodal-domain count ratio vs phi (Penrose-tiling analogy)")
print("=" * 65)

k0 = 2.0
def make_substrate(span, L):
    x = np.linspace(-span / 2, span / 2, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    for th in thetas:
        field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))
    return field

spans_and_L = [(20, 300), (40, 600), (80, 1200), (160, 2400)]

print(f"{'span':>6s} {'L':>6s} {'#pos domains':>13s} {'#neg domains':>13s} {'ratio':>8s}")
print("-" * 55)

ratios = []
for span, L in spans_and_L:
```

```
    field = make_substrate(span, L)
    pos_mask = field > 0
    neg_mask = field < 0
    lbl_pos, n_pos = ndimage.label(pos_mask)
    lbl_neg, n_neg = ndimage.label(neg_mask)

    ratio_pn = n_pos / n_neg if n_neg else float('nan')
    ratios.append(ratio_pn)
    print(f"{span:6d} {L:6d} {n_pos:13d} {n_neg:13d} {ratio_pn:8.4f}")

print("-" * 55)
print(f"{'phi':>39s} {phi:8.4f}")

# ---------------------------------------------------------------------
# Also check domain AREA ratio (total positive area / total negative area) rather than just
# domain count, as a second way of looking for phi.
# ---------------------------------------------------------------------

print("\nArea-based check (total positive area / total negative area):")
for span, L in spans_and_L:
    field = make_substrate(span, L)
    pos_frac = (field > 0).mean()
    neg_frac = (field < 0).mean()
    area_ratio = pos_frac / neg_frac if neg_frac else float('nan')
    print(f"  span={span:4d}: area ratio = {area_ratio:.4f}")
```

correlated_percolation_test.py

```
"""
correlated_percolation_test.py

The Q-Brane field is a superposition of many plane waves at a single wavenumber with random
directions and phases. This exact construction is what's called the "random wave model" (Berry's
random plane waves), used to model chaotic quantum wavefunctions. Its level-set (nodal
domain) percolation has a specific, known theoretical result: Bogomolny & Schmit, "Percolation
model for nodal domains of chaotic wave functions", PRL 88, 114102 (2002): despite the field
being smooth and long-range correlated (NOT independent-site disorder), thresholding it at the
ZERO (median) level produces a percolation transition statistically indistinguishable from
standard 2D independent-site percolation -- same critical exponents (tau = 187/91, D = 91/48).

This is expected to hold specifically at t=0 (a symmetry point for a field with zero mean). Away
from t=0, there's no such guarantee and exponents are expected to drift -- which is exactly the
anomaly the previous test found by accident (it located its "critical" threshold near t=-0.014,
close to but not exactly zero, using a fairly small L=300 single-realization estimate).
```

This script does two things properly:

A) Pools many realizations at a much larger grid size, thresholds EXACTLY at t=0, and remeasures tau and D against standard percolation values -- the theoretically correct test.

B) Runs a real finite-size scaling analysis (multiple system sizes) to extract an actual correlation-length exponent nu, rather than the earlier single-run rough estimate.

"""

```
import numpy as np
from scipy import ndimage

k0 = 2.0

def make_wave_field(L, seed_rng, n_waves=16):
    x = np.linspace(-10, 10, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

# ----------------------------------------------------------------------
# PART A -- exponents AT THE ZERO LEVEL (the theoretically correct test)
# ----------------------------------------------------------------------

print("=" * 70)
print("PART A: nodal-domain (t=0) percolation exponents")
print("=" * 70)

rng = np.random.default_rng(123)
L_big = 500
N_REAL_A = 30

pooled_nongiant_sizes = []
D_estimates = []

for r in range(N_REAL_A):
    field = make_wave_field(L_big, rng)
    mask = field > 0.0  # the theoretically motivated threshold
    lbl, n = ndimage.label(mask)
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
```

```
    giant_idx = np.argmax(sizes)
    non_giant = np.delete(sizes, giant_idx)
    pooled_nongiant_sizes.extend(non_giant[non_giant >= 2].tolist())

    giant_mask = lbl == (giant_idx + 1)

    def box_count_dimension(m, min_box=2, max_box=None):
        if max_box is None:
            max_box = min(m.shape) // 4
        box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
        counts = []
        for bs in box_sizes:
            ny, nx = m.shape[0] // bs, m.shape[1] // bs
            trimmed = m[:ny * bs, :nx * bs]
            blocks = trimmed.reshape(ny, bs, nx, bs)
            occ = blocks.any(axis=(1, 3))
            counts.append(occ.sum())
        counts = np.array(counts)
        valid = counts > 0
        lx = np.log(1.0 / box_sizes[valid])
        ly = np.log(counts[valid])
        slope, _ = np.polyfit(lx, ly, 1)
        return slope

    D_estimates.append(box_count_dimension(giant_mask))

sizes_arr = np.array(pooled_nongiant_sizes, dtype=float)
xmin = 2.0
n = len(sizes_arr)
tau_hat = 1 + n / np.sum(np.log(sizes_arr / xmin))
tau_stderr = (tau_hat - 1) / np.sqrt(n)

D_mean, D_sem = np.mean(D_estimates), np.std(D_estimates) / np.sqrt(len(D_estimates))

print(f"{N_REAL_A} realizations at L={L_big}, thresholded at t=0\n")
print(f"tau (measured)  = {tau_hat:.3f} +/- {tau_stderr:.3f}   (n={n} pooled clusters)")
print(f"tau (theory)    = {187/91:.4f}  [standard 2D percolation]")
print(f"  -> {abs(tau_hat - 187/91)/tau_stderr:.1f} sigma from theory\n")

print(f"D (measured)    = {D_mean:.3f} +/- {D_sem:.3f}")
print(f"D (theory)      = {91/48:.4f}  [standard 2D percolation]")
print(f"  -> {abs(D_mean - 91/48)/D_sem:.1f} sigma from theory\n")
```

```
# ---------------------------------------------------------------------
# PART B -- finite-size scaling for a real correlation-length exponent nu
# ---------------------------------------------------------------------

print("=" * 70)
print("PART B: finite-size scaling for nu (pseudo-critical point spread)")
print("=" * 70)
print("Method: at each system size L, find each realization's own")
print("'pseudo-critical' threshold (where the susceptibility peaks), then")
print("measure how much that pseudo-critical point varies realization to")
print("realization. That spread shrinks with system size as L^(-1/nu) --")
print("the standard finite-size-scaling estimator for nu.\n")
L_values = [80, 120, 160, 220, 300]
N_REAL_B = 18
window = 1.2  # search thresholds in [-window, window] around 0
n_thresh_b = 41

pseudo_crit_spread = []

for L in L_values:
    rng_b = np.random.default_rng(7000 + L)
    pseudo_crits = []
    thresholds = np.linspace(-window, window, n_thresh_b)
    for r in range(N_REAL_B):
        field = make_wave_field(L, rng_b)
        suscept = np.zeros(n_thresh_b)
        for i, t in enumerate(thresholds):
            mask = field > t
            lbl, nlab = ndimage.label(mask)
            if nlab == 0:
                continue
            sizes = ndimage.sum(mask, lbl, index=range(1, nlab + 1))
            non_giant = np.delete(sizes, np.argmax(sizes))
            suscept[i] = non_giant.mean() if len(non_giant) else 0.0
        pseudo_crits.append(thresholds[np.argmax(suscept)])
    spread = np.std(pseudo_crits)
    pseudo_crit_spread.append(spread)
    print(f"  L={L:4d}: pseudo-critical spread = {spread:.4f}")

logL = np.log(L_values)
logspread = np.log(pseudo_crit_spread)
slope, intercept = np.polyfit(logL, logspread, 1)
nu_hat = -1 / slope
```

```
print(f"\nfit: log(spread) = {slope:.3f} * log(L) + const")
print(f"=> nu (measured) = {nu_hat:.3f}")
print(f"   nu (theory)   = 4/3 = {4/3:.4f}  [standard 2D percolation]")
print(f"   relative deviation: {abs(nu_hat - 4/3)/(4/3)*100:.1f}%")
domain_size_convergence.py

"""
domain_size_convergence.py

The wave-count test ruled out "not enough directions" as the reason the Q-Brane field's nodal-domain percolation exponents don't match standard 2D percolation. The more likely culprit: the field's correlation length is fixed by its wavenumber (k0), not by wave count, and the domain used so far (span 20, wavelength ~2.4) is only about 8 wavelengths across -- far too small a box, in correlation-length units, for exponents to have converged to their infinite-system values.

This test holds the wavelength fixed and grows the domain span, while keeping grid resolution per wavelength constant (so aliasing doesn't change field character across runs), and checks whether tau and D drift toward the standard 2D percolation values (187/91, 91/48) as the domain-to-wavelength ratio grows.

"""
import numpy as np
from scipy import ndimage

k0 = 2.0
wavelength = 2 * np.pi / (1.3 * k0)
n_waves = 32  # fixed; shown not to matter in the previous test

POINTS_PER_WAVELENGTH = 12
dx = wavelength / POINTS_PER_WAVELENGTH

def make_wave_field(span, seed_rng, n_waves=n_waves):
    L = int(span / dx)
    x = np.linspace(-span / 2, span / 2, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves), L

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
```

```
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope

spans = [20, 40, 80, 160, 320]
n_realizations = [24, 18, 12, 8, 5]

tau_theory = 187 / 91
D_theory = 91 / 48

print(f"wavelength = {wavelength:.3f}, grid spacing = {dx:.3f}\n")
print(f"{'span':>6s} {'~wavelengths':>13s} {'L':>6s} {'tau':>8s} {'+/-':>7s} {'D':>8s}
{'+/-':>7s}")
print("-" * 65)

results = []
for span, n_real in zip(spans, n_realizations):
    rng = np.random.default_rng(2000 + span)
    pooled_sizes = []
    D_estimates = []
    L_used = None
    for r in range(n_real):
        field, L_used = make_wave_field(span, rng)
        mask = field > 0.0
        lbl, n = ndimage.label(mask)
        sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
        giant_idx = np.argmax(sizes)
        non_giant = np.delete(sizes, giant_idx)
        pooled_sizes.extend(non_giant[non_giant >= 2].tolist())
        giant_mask = lbl == (giant_idx + 1)
        D_estimates.append(box_count_dimension(giant_mask))
```

```
    sizes_arr = np.array(pooled_sizes, dtype=float)
    xmin = 2.0
    n_pooled = len(sizes_arr)
    tau_hat = 1 + n_pooled / np.sum(np.log(sizes_arr / xmin))
    tau_stderr = (tau_hat - 1) / np.sqrt(n_pooled)
    D_mean = np.mean(D_estimates)
    D_sem = np.std(D_estimates) / np.sqrt(len(D_estimates))

    n_wavelengths = span / wavelength
    Results:append((span, n_wavelengths, L_used, tau_hat, tau_stderr, D_mean, D_sem))
    print(f"{span:6d} {n_wavelengths:13.1f} {L_used:6d} {tau_hat:8.3f} {tau_stderr:7.3f} "
          f"{D_mean:8.3f} {D_sem:7.3f}")

print("-" * 65)
print(f"{'theory':>27s} {tau_theory:8.3f} {':>7s} {D_theory:8.3f}")

print("\nTrend (deviation from theory vs domain-to-wavelength ratio):")
for span, n_wl, L_used, tau_hat, tau_se, D_mean, D_se in results:
    print(f"  {n_wl:6.1f} wavelengths across:  tau_dev={tau_hat - tau_theory:+.3f}   "
          f"D_dev={D_mean - D_theory:+.3f}")
resolution_convergence.py

"""
resolution_convergence.py

Domain size (wavelengths spanned) is now ruled out as the explanation for tau's stubborn
mismatch with standard percolation. The remaining confound: pixel resolution *within* a
wavelength. At only 12 grid points per wavelength, small clusters (a handful of pixels) could be
dominated by discretization noise rather than real geometry, which would bias the MLE tau
estimate.

This holds the domain size fixed (span = 80, ~33 wavelengths – large enough per the previous
test) and increases grid points per wavelength, to see whether tau and D drift as pixel-level
discretization noise is reduced.

"""
import numpy as np
from scipy import ndimage

k0 = 2.0
wavelength = 2 * np.pi / (1.3 * k0)
n_waves = 32
span = 80.0  # ~33 wavelengths, shown to be past the domain-size plateau
```

```
def make_wave_field(L, seed_rng, n_waves=n_waves):
    x = np.linspace(-span / 2, span / 2, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope

points_per_wavelength = [6, 12, 24, 48, 96]
n_realizations = [20, 15, 10, 6, 3]

tau_theory = 187 / 91
D_theory = 91 / 48

print(f"span={span} (~{span/wavelength:.1f} wavelengths, fixed)\n")
print(f"{'ppw':>5s} {'L':>6s} {'xmin_px':>8s} {'tau':>8s} {'+/-':>7s} {'D':>8s} {'+/-':>7s}")
print("-" * 55)

results = []
for ppw, n_real in zip(points_per_wavelength, n_realizations):
    dx = wavelength / ppw
    L = int(span / dx)
rng = np.random.default_rng(3000 + ppw)
```

```
# ---------------------------------------------------------------------
# xmin in pixel-count terms should scale with resolution too -- a "2-pixel cluster" at low
# resolution is a much bigger physical object than a "2-pixel cluster" at high resolution, so to
# compare like with like, set xmin as a fixed physical area (in units of wavelength^2) converted
# to pixel count at this resolution.
# ---------------------------------------------------------------------

    xmin_physical_area = 0.15 * wavelength ** 2
    xmin_px = max(2.0, xmin_physical_area / (dx ** 2))

    pooled_sizes = []
    D_estimates = []
    for r in range(n_real):
        field = make_wave_field(L, rng)
        mask = field > 0.0
        lbl, n = ndimage.label(mask)
        sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
        giant_idx = np.argmax(sizes)
        non_giant = np.delete(sizes, giant_idx)
        pooled_sizes.extend(non_giant[non_giant >= xmin_px].tolist())
        giant_mask = lbl == (giant_idx + 1)
        D_estimates.append(box_count_dimension(giant_mask))

    sizes_arr = np.array(pooled_sizes, dtype=float)
    n_pooled = len(sizes_arr)
    tau_hat = 1 + n_pooled / np.sum(np.log(sizes_arr / xmin_px))
    tau_stderr = (tau_hat - 1) / np.sqrt(n_pooled)
    D_mean = np.mean(D_estimates)
    D_sem = np.std(D_estimates) / np.sqrt(len(D_estimates))

    Results:append((ppw, L, xmin_px, tau_hat, tau_stderr, D_mean, D_sem))
    print(f"{ppw:5d} {L:6d} {xmin_px:8.1f} {tau_hat:8.3f} {tau_stderr:7.3f} "
          f"{D_mean:8.3f} {D_sem:7.3f}")

print("-" * 55)
print(f"{'theory':>20s} {tau_theory:8.3f} {":>7s} {D_theory:8.3f}")

print("\nTrend (deviation from theory vs grid points per wavelength):")
for ppw, L, xmin_px, tau_hat, tau_se, D_mean, D_se in results:
print(f"  ppw={ppw:3d}:  tau_dev={tau_hat - tau_theory:+.3f}   D_dev={D_mean – D_theory:
+.3f}")

wave_count_convergence.py
```

```
"""
wave_count_convergence.py

Tests whether the Q-Brane field's nodal-domain percolation exponents drift toward the standard
2D percolation values (tau=187/91, D=91/48) as the number of superposed plane waves
increases toward the continuum random-wave-field limit that Bogomolny & Schmit's result
actually applies to.

Same measurement as before (threshold exactly at t=0, pooled cluster sizes -> MLE tau, box-
counting -> D), repeated at increasing n_waves.

"""
import numpy as np
from scipy import ndimage

k0 = 2.0

def make_wave_field(L, seed_rng, n_waves):
    x = np.linspace(-10, 10, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope
```

```
L = 350
N_REAL = 18
wave_counts = [16, 32, 64, 128, 256, 512]

tau_theory = 187 / 91
D_theory = 91 / 48

print(f"{'n_waves':>8s} {'tau':>8s} {'+/-':>7s} {'D':>8s} {'+/-':>7s} {'n_clusters':>11s}")
print("-" * 55)

results = []
for n_waves in wave_counts:
    rng = np.random.default_rng(1000 + n_waves)
    pooled_sizes = []
    D_estimates = []
    for r in range(N_REAL):
        field = make_wave_field(L, rng, n_waves)
        mask = field > 0.0
        lbl, n = ndimage.label(mask)
        sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
        giant_idx = np.argmax(sizes)
        non_giant = np.delete(sizes, giant_idx)
        pooled_sizes.extend(non_giant[non_giant >= 2].tolist())
        giant_mask = lbl == (giant_idx + 1)
        D_estimates.append(box_count_dimension(giant_mask))

    sizes_arr = np.array(pooled_sizes, dtype=float)
    xmin = 2.0
    n_pooled = len(sizes_arr)
    tau_hat = 1 + n_pooled / np.sum(np.log(sizes_arr / xmin))
    tau_stderr = (tau_hat - 1) / np.sqrt(n_pooled)
    D_mean = np.mean(D_estimates)
    D_sem = np.std(D_estimates) / np.sqrt(len(D_estimates))

    Results:append((n_waves, tau_hat, tau_stderr, D_mean, D_sem, n_pooled))
    print(f"{n_waves:8d} {tau_hat:8.3f} {tau_stderr:7.3f} {D_mean:8.3f} {D_sem:7.3f}
{n_pooled:11d}")
print("-" * 55)
print(f"{'theory':>8s} {tau_theory:8.3f} {'':>7s} {D_theory:8.3f}")

print("\nTrend check (is deviation from theory shrinking as n_waves grows?):")
for n_waves, tau_hat, tau_se, D_mean, D_se, _ in results:
    tau_dev = tau_hat - tau_theory
    D_dev = D_mean - D_theory
```

```
    print(f"  n_waves={n_waves:4d}:  tau_dev={tau_dev:+.3f}   D_dev={D_dev:+.3f}")
```

symmetry_vs_asymmetry.py

```
"""
symmetry_vs_asymmetry.py

Every percolation test so far used fully random (asymmetric) wave directions -- matching the Q-
Brane stage's original construction. The Substrate stage, by contrast, uses exactly evenly-spaced
directions (5-fold symmetric). This test isolates symmetry itself as a variable: same wave count,
same domain size, same resolution -- only the direction distribution changes (evenly spaced vs
uniformly random). Phases are randomized in both cases either way (that's the "disorder"
being percolated against), only the DIRECTIONS are controlled.

"""
import numpy as np
from scipy import ndimage

k0 = 2.0
wavelength = 2 * np.pi / (1.3 * k0)
span = 80.0
ppw = 24
dx = wavelength / ppw
L = int(span / dx)

tau_theory = 187 / 91
D_theory = 91 / 48

def make_field(seed_rng, n_waves, symmetric):
    x = np.linspace(-span / 2, span / 2, L)
    X, Y = np.meshgrid(x, x)
    field = np.zeros((L, L))
    if symmetric:
        dirs = np.array([2 * np.pi * j / n_waves for j in range(n_waves)])
    else:
        dirs = seed_rng.uniform(0, 2 * np.pi, n_waves)
    phases = seed_rng.uniform(0, 2 * np.pi, n_waves)  # disorder in both cases
    for th, ph in zip(dirs, phases):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

def box_count_dimension(m, min_box=2, max_box=None):
    if max_box is None:
        max_box = min(m.shape) // 4
    box_sizes = np.unique(np.geomspace(min_box, max_box, 12).astype(int))
```

```
    counts = []
    for bs in box_sizes:
        ny, nx = m.shape[0] // bs, m.shape[1] // bs
        trimmed = m[:ny * bs, :nx * bs]
        blocks = trimmed.reshape(ny, bs, nx, bs)
        occ = blocks.any(axis=(1, 3))
        counts.append(occ.sum())
    counts = np.array(counts)
    valid = counts > 0
    lx = np.log(1.0 / box_sizes[valid])
    ly = np.log(counts[valid])
    slope, _ = np.polyfit(lx, ly, 1)
    return slope

xmin_physical_area = 0.15 * wavelength ** 2
xmin_px = max(2.0, xmin_physical_area / (dx ** 2))

n_waves_list = [5, 16, 32]
n_real = 16

print(f"span={span} (~{span/wavelength:.1f} wavelengths), ppw={ppw}, L={L}\n")
print(f"{'n_waves':>8s} {'symmetry':>10s} {'tau':>8s} {'+/-':>7s} {'D':>8s} {'+/-':>7s}")
print("-" * 55)

results = {}
for n_waves in n_waves_list:
    for symmetric in [True, False]:
        rng = np.random.default_rng(4000 + n_waves + (1000 if symmetric else 0))
        pooled_sizes = []
        D_estimates = []
        for r in range(n_real):
            field = make_field(rng, n_waves, symmetric)
            mask = field > 0.0
            lbl, n = ndimage.label(mask)
            sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
            giant_idx = np.argmax(sizes)
            non_giant = np.delete(sizes, giant_idx)
            pooled_sizes.extend(non_giant[non_giant >= xmin_px].tolist())
            giant_mask = lbl == (giant_idx + 1)
            D_estimates.append(box_count_dimension(giant_mask))

        sizes_arr = np.array(pooled_sizes, dtype=float)
        n_pooled = len(sizes_arr)
        tau_hat = 1 + n_pooled / np.sum(np.log(sizes_arr / xmin_px))
```

```
        tau_stderr = (tau_hat - 1) / np.sqrt(n_pooled)
        D_mean = np.mean(D_estimates)
        D_sem = np.std(D_estimates) / np.sqrt(len(D_estimates))

        label = "symmetric" if symmetric else "random"
        results[(n_waves, symmetric)] = (tau_hat, tau_stderr, D_mean, D_sem)
        print(f"{n_waves:8d} {label:>10s} {tau_hat:8.3f} {tau_stderr:7.3f} "
              f"{D_mean:8.3f} {D_sem:7.3f}")

print("-" * 55)
print(f"{'theory':>19s} {tau_theory:8.3f} {':>7s} {D_theory:8.3f}")

print("\nSymmetric vs random, same n_waves:")
for n_waves in n_waves_list:
    tau_s, _, D_s, _ = results[(n_waves, True)]
    tau_a, _, D_a, _ = results[(n_waves, False)]
    print(f"  n_waves={n_waves:3d}:  tau (sym-rand)={tau_s - tau_a:+.3f}    "
          f"D (sym-rand)={D_s - D_a:+.3f}")
```

C.7 - Emergence Pipeline - Corrected border_value Fix (B.14 / A.4)

emergence_pipeline_corrected.py

```
"""
emergence_pipeline.py

A formal-analogy simulation of the five-stage progression described in the manuscript's
cosmological framework:

    Null Point -> Quasi-Crystalline Substrate -> Quantum Foam -> Q-Brane -> Universe
(Ignition of Physical Reality)

This is NOT a physical model of the actual universe. It is a mathematical illustration chosen
because each stage admits a well-defined, computable notion of "order" or "coherence," and the
five values chain together into a single curve that has the right qualitative shape for the narrative:
zero -> latent order -> decoherence -> a sharp percolation-type transition ("ignition") -> a stable
plateau.

Two different but complementary order parameters are used, because no single classical statistic
is meaningful across all five stages:

  * Spectral coherence (Psi) -- how sharply peaked a field's power spectrum is. Meaningful for
stages with wave-like / quasiperiodic structure (Substrate, Foam).
```

*** Connective coherence (Phi)** -- the fraction of the domain occupied by the single largest connected region once a field is thresholded. Meaningful for stages defined by the emergence of large-scale connected structure (Q-Brane, Universe).

Both are normalised to [0, 1] so they can sit on one axis for the summary plot. The Null Point is defined analytically as the zero of both (no differentiated structure exists prior to it).

"""

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage

rng = np.random.default_rng(7)

L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)

# ----------------------------------------------------------------------
# Stage 0 -- Null Point
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Perfect, undifferentiated symmetry. No broken symmetry means no measurable structure exists
# yet -- coherence is undefined/zero by construction, not computed from a field.
# ----------------------------------------------------------------------

C0 = 0.0

# ----------------------------------------------------------------------
# Stage 1 -- Quasi-Crystalline Substrate
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Superposition of five plane waves at 72-degree increments. Five-fold symmetric wave sets
# cannot tile a plane periodically, so the resulting density field is quasiperiodic: long-range order
# without translational periodicity -- the formal analogue of a quasi-crystalline substrate.
# ----------------------------------------------------------------------

k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])
substrate_field = np.zeros((L, L))
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))
```

```
def peak_to_mean_ratio(field):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    cy, cx = np.array(P.shape) // 2
    P[cy - 2:cy + 3, cx - 2:cx + 3] = 0  # zero out the DC peak/its neighborhood
return P.max() / P.mean()

# ---------------------------------------------------------------------
# Calibrate the [0,1] spectral-coherence scale against two fixed anchors: pure white noise (the
# least-ordered field possible on this grid) at the bottom, and the noiseless quasicrystal substrate
# (the most-ordered field used in this pipeline) at the top. Everything else is placed on a log scale
# between them, since peak-to-mean ratios span several orders of magnitude.
# ---------------------------------------------------------------------

_noise_floor_ratio = peak_to_mean_ratio(rng.normal(size=(L, L)))
_substrate_ratio = peak_to_mean_ratio(substrate_field)
_log_lo, _log_hi = np.log(_noise_floor_ratio), np.log(_substrate_ratio)

def spectral_coherence(field):
    ratio = peak_to_mean_ratio(field)
    return float(np.clip((np.log(ratio) - _log_lo) / (_log_hi - _log_lo), 0, 1))

Psi_substrate = spectral_coherence(substrate_field)

# ---------------------------------------------------------------------
# Stage 2 -- Quantum Foam
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# The same five-wave construction, but now each realization gets random phases, random
# amplitude jitter, and additive white noise -- a seething, fluctuating field rather than a static one.
# Averaging the *power spectrum* (not the field) over many independent realizations shows the
# same underlying wavevectors, but broadened and buried in noise: order is present in principle
# but not resolved in any single instant. This is what should decohere the substrate's sharp peaks.
# ---------------------------------------------------------------------

phases = rng.uniform(0, 2 * np.pi, size=5)
amps = 1 + 0.35 * rng.normal(size=5)
foam_signal = np.zeros((L, L))
for th, ph, a in zip(thetas, phases, amps):
foam_signal += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)

# ---------------------------------------------------------------------
# Coherence is measured on a single instantaneous realization, not an ensemble average --
```

```
# averaging over many noisy snapshots would recover the underlying wavevectors and defeat the
# point. At any given "moment," strong stochastic noise buries the substrate's order beneath
# foam-like fluctuations, which is what should knock the coherence index down from the
# substrate stage.
# ---------------------------------------------------------------------

foam_snapshot = foam_signal + 8.0 * rng.normal(size=(L, L))
Psi_foam = spectral_coherence(foam_snapshot)

# ---------------------------------------------------------------------
# Stage 3 -- Q-Brane
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# Many plane waves (16) at random directions and random phases, superposed and thresholded.
# As the threshold is swept, the occupied region undergoes a percolation transition: below
# threshold, occupied cells form small disconnected islands; above a critical value, a single
# connected component suddenly spans a large fraction of the domain. That sudden jump is the
# formal analogue of "ignition" -- a coherent, connected q-brane condensing out of an incoherent
# field.
# ---------------------------------------------------------------------

n_waves = 16
qbrane_field = np.zeros((L, L))
dirs = rng.uniform(0, 2 * np.pi, n_waves)
phases_q = rng.uniform(0, 2 * np.pi, n_waves)
for th, ph in zip(dirs, phases_q):
    qbrane_field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
qbrane_field /= np.sqrt(n_waves)

thresholds = np.linspace(qbrane_field.min(), qbrane_field.max(), 80)
giant_frac = []
for t in thresholds:
    mask = qbrane_field > t
    lbl, n = ndimage.label(mask)
    if n == 0:
        giant_frac.append(0.0)
        continue
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
    giant_frac.append(sizes.max() / mask.size)
giant_frac = np.array(giant_frac)

# ---------------------------------------------------------------------
# `thresholds` runs low -> high, so `giant_frac` runs *high -> low* (a low threshold admits
```

```
# almost the whole field trivially; that is not a meaningful "structure," just an unselective mask).
# The meaningful direction is a high, selective threshold (sparse, disconnected islands) relaxing
# down toward a lower one. Two points on that sweep matter here:
#   t_onset  -- the sharp jump where a giant cluster first spans >20% of
#               the domain (the "ignition" instant itself)
#   t_stable -- a further, more relaxed point (>75%) where the ignited
#               structure has fully connected and stopped changing
#               qualitatively -- this is the Q-Brane's realized state,
#               not just its onset, so it is the value carried into the
#               coherence curve.
# ---------------------------------------------------------------------

rev_frac = giant_frac[::-1]
rev_thresh = thresholds[::-1]

onset_hits = np.where(rev_frac > 0.20)[0]
onset_idx = onset_hits[0] if len(onset_hits) else int(np.argmax(rev_frac))
t_onset = rev_thresh[onset_idx]

stable_hits = np.where(rev_frac > 0.75)[0]
stable_idx = stable_hits[0] if len(stable_hits) else int(np.argmax(rev_frac))
t_crit = rev_thresh[stable_idx]

mask_crit = qbrane_field > t_crit
lbl_crit, n_crit = ndimage.label(mask_crit)
sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
giant_label = int(np.argmax(sizes_crit)) + 1
giant_mask = lbl_crit == giant_label

Phi_qbrane = rev_frac[stable_idx]

# ---------------------------------------------------------------------
# Stage 4 -- Universe (Ignition of Physical Reality)
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# The percolating giant cluster is morphologically closed (small gaps filled) and lightly
# smoothed -- representing the connected structure settling into a stable configuration rather than
# a marginal, noisy one. The connective coherence here should sit at or slightly above the q-
# brane value and, importantly, hold steady rather than collapse back down -- a plateau rather
# than a spike.
# ---------------------------------------------------------------------

universe_mask = ndimage.binary_closing(giant_mask, iterations=2, border_value=1)
```

```
# ----------------------------------------------------------------------
# NOTE (correction per B.14 robustness check): scipy's binary_closing default, border_value=0,
# is not guaranteed extensive at the array edges on a finite grid -- it can remove area near the
# border rather than only adding it, which is a boundary-handling artifact rather than a real effect.
# Classical morphological closing is provably extensive (closed area >= original area) when
# border_value=1 is used, which is the correct setting for this step.
# ----------------------------------------------------------------------

universe_field = ndimage.gaussian_filter(universe_mask.astype(float), sigma=1.2)
Phi_universe = universe_mask.mean()

# ----------------------------------------------------------------------
# Summary
# ----------------------------------------------------------------------

stages = ["Null\nPoint", "Quasi-Crystalline\nSubstrate", "Quantum\nFoam",
          "Q-Brane", "Universe\n(Ignition)"]
values = [C0, Psi_substrate, Psi_foam, Phi_qbrane, Phi_universe]

print("Stage coherence values:")
for s, v in zip(stages, values):
    print(f"  {s.replace(chr(10), ' '):28s} {v:.4f}")
print(f"\nPercolation ignition onset threshold   = {t_onset:.4f}  (giant cluster crosses 20%)")
print(f"Post-ignition stable threshold         = {t_crit:.4f}  (giant cluster crosses 75%)")
print(f"Q-Brane giant-cluster fraction          = {Phi_qbrane:.4f}")
print(f"Stabilized universe fraction           = {Phi_universe:.4f}")

# ----------------------------------------------------------------------
# Figure
# ----------------------------------------------------------------------

fig = plt.figure(figsize=(15, 9))
gs = fig.add_gridspec(2, 5, height_ratios=[1, 0.9])
extent = [x.min(), x.max(), x.min(), x.max()]

ax0 = fig.add_subplot(gs[0, 0])
ax0.set_facecolor('black')
ax0.plot(0, 0, 'o', color='white', markersize=6)
ax0.set_xlim(-1, 1); ax0.set_ylim(-1, 1)
ax0.set_title("Null Point", fontsize=10)
ax0.set_xticks([]); ax0.set_yticks([])

ax1 = fig.add_subplot(gs[0, 1])
```

```
ax1.imshow(substrate_field, extent=extent, cmap='cividis')
ax1.set_title("Quasi-Crystalline\nSubstrate", fontsize=10)
ax1.set_xticks([]); ax1.set_yticks([])

ax2 = fig.add_subplot(gs[0, 2])
ax2.imshow(foam_snapshot, extent=extent, cmap='magma')
ax2.set_title("Quantum Foam\n(single realization)", fontsize=10)
ax2.set_xticks([]); ax2.set_yticks([])
ax3 = fig.add_subplot(gs[0, 3])
disp = np.zeros((L, L, 3))
disp[..., 2] = 0.15 * (qbrane_field > qbrane_field.min())
disp[giant_mask] = [1.0, 0.75, 0.1]
ax3.imshow(disp, extent=extent)
ax3.set_title(f"Q-Brane\n(giant cluster, t={t_crit:.2f})", fontsize=10)
ax3.set_xticks([]); ax3.set_yticks([])

ax4 = fig.add_subplot(gs[0, 4])
ax4.imshow(universe_field, extent=extent, cmap='inferno')
ax4.set_title("Universe\n(stabilized structure)", fontsize=10)
ax4.set_xticks([]); ax4.set_yticks([])

ax5 = fig.add_subplot(gs[1, :])
xs = np.arange(5)
ax5.plot(xs, values, '-o', color='#c9963d', linewidth=2.5, markersize=9,
        markerfacecolor='#1a1a2e', markeredgecolor='#c9963d', markeredgewidth=2)
ax5.fill_between(xs, values, color='#c9963d', alpha=0.12)
ax5.set_xticks(xs)
ax5.set_xticklabels(stages, fontsize=10)
ax5.set_ylabel("Coherence index  (0 = undifferentiated, 1 = fully coherent)")
ax5.set_ylim(-0.05, 1.05)
ax5.set_title("Structural coherence across the five stages", fontsize=12)
ax5.grid(alpha=0.25)
ax5.axvspan(2.5, 3.5, color='#c9963d', alpha=0.08)
ax5.annotate("percolation-type\nignition", xy=(3, values[3]),
            xytext=(3.15, max(values) * 0.55),
            fontsize=9, ha='left',
            arrowprops=dict(arrowstyle='->', color='gray'))

fig.suptitle("Formal-Analogy Simulation: Null Point \u2192 Substrate \u2192 Foam \u2192 Q-
Brane \u2192 Universe",
            fontsize=13, y=0.99)
fig.text(0.5, 0.965,
        "Note: Foam uses spectral coherence (Psi); Q-Brane/Universe use connective coherence
(Phi) -- "
```

```
        "the Foam\u2192Q-Brane step is a metric switch, not a single continuously-evolving
quantity.",
        ha='center', fontsize=8, style='italic', color='#555555')
fig.tight_layout(rect=[0, 0, 1, 0.94])
fig.savefig("/home/claude/sims/A4a_emergence_pipeline_corrected.png", dpi=180,
facecolor='white')
print("\nSaved figure to A4a_emergence_pipeline_corrected.png")
```

C.8 - Foam and Universe Robustness Sweep (B.14)

B14_foam_universe_robustness.py

```
"""
B14_foam_universe_robustness.py

Robustness / parameter-sweep check on the Quantum Foam and Universe stages of A.4
(emergence_pipeline.py). This does NOT introduce new standalone simulations for these two
stages -- it re-uses A.4's exact constructions and asks whether the two qualitative claims A.4
makes about them survive being re-run under many seeds and parameter choices, rather than the
one seed A.4 happens to use.

Claim 1 (Quantum Foam): a noisy single-realization snapshot of the five-wave substrate has
LOWER spectral coherence than the clean substrate -- i.e. noise reliably decoheres the signal,
this isn't a one-off artifact of rng seed 7.

Claim 2 (Universe): the morphologically-closed, lightly-smoothed giant cluster sits AT OR
ABOVE the raw percolated Q-Brane fraction and holds roughly steady across reasonable choices
of closing/smoothing parameters -- i.e. it's a genuine plateau, not a fragile one-seed coincidence.

CORRECTION (post-hoc): the first version of this sweep used scipy's binary_closing default,
border_value=0, which is NOT guaranteed to be extensive (superset-preserving) at the edges of a
finite array -- it can remove area near the border rather than only adding it, purely as a boundary-
handling artifact. That produced a spurious small NEGATIVE delta that read as "Universe sits
below Q-Brane." Classical morphological closing is provably extensive (closed area >= original
area) when border_value=1 is used instead, which is the correct setting and is what this version
uses.

Method: repeat each measurement across many independent seeds, and for Claim 2 also sweep
the post-processing parameters (closing iterations, smoothing sigma) that A.4 fixed at single
values (2, 1.2). Report the distribution, not just a mean, and state plainly whether each claim
holds up.

"""
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy import ndimage

L = 200

# ---------------------------------------------------------------------
# slightly smaller than A.4's 300 for sweep speed; verified this does not change the qualitative
# outcome (see note at bottom)
# ---------------------------------------------------------------------

x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)
k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])

# ---------------------------------------------------------------------
# Shared machinery (identical in spirit to A.4)
# ---------------------------------------------------------------------
substrate_field = np.zeros((L, L))
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))

def peak_to_mean_ratio(field):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    cy, cx = np.array(P.shape) // 2
    P[cy - 2:cy + 3, cx - 2:cx + 3] = 0
    return P.max() / P.mean()

_calib_rng = np.random.default_rng(0)
_noise_floor_ratio = peak_to_mean_ratio(_calib_rng.normal(size=(L, L)))
_substrate_ratio = peak_to_mean_ratio(substrate_field)
_log_lo, _log_hi = np.log(_noise_floor_ratio), np.log(_substrate_ratio)

def spectral_coherence(field):
    ratio = peak_to_mean_ratio(field)
    return float(np.clip((np.log(ratio) - _log_lo) / (_log_hi - _log_lo), 0, 1))

Psi_substrate = spectral_coherence(substrate_field)

def make_foam_snapshot(seed, noise_amp=8.0, jitter=0.35):
    rng = np.random.default_rng(seed)
    phases = rng.uniform(0, 2 * np.pi, size=5)
    amps = 1 + jitter * rng.normal(size=5)
    foam_signal = np.zeros((L, L))
```

```
    for th, ph, a in zip(thetas, phases, amps):
        foam_signal += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return foam_signal + noise_amp * rng.normal(size=(L, L))

def make_qbrane_field(seed, n_waves=16):
    rng = np.random.default_rng(seed)
    dirs = rng.uniform(0, 2 * np.pi, n_waves)
    phases_q = rng.uniform(0, 2 * np.pi, n_waves)
    field = np.zeros((L, L))
    for th, ph in zip(dirs, phases_q):
        field += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
    return field / np.sqrt(n_waves)

def qbrane_giant_cluster(field):
    thresholds = np.linspace(field.min(), field.max(), 80)
    giant_frac = []
    for t in thresholds:
        mask = field > t
        lbl, n = ndimage.label(mask)
        if n == 0:
            giant_frac.append(0.0)
            continue
        sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
        giant_frac.append(sizes.max() / mask.size)
    giant_frac = np.array(giant_frac)
    rev_frac = giant_frac[::-1]
    rev_thresh = thresholds[::-1]
    stable_hits = np.where(rev_frac > 0.75)[0]
    stable_idx = stable_hits[0] if len(stable_hits) else int(np.argmax(rev_frac))
    t_crit = rev_thresh[stable_idx]
    mask_crit = field > t_crit
    lbl_crit, n_crit = ndimage.label(mask_crit)
    sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
    giant_label = int(np.argmax(sizes_crit)) + 1
    giant_mask = lbl_crit == giant_label
    Phi_qbrane = rev_frac[stable_idx]
    return giant_mask, Phi_qbrane

# ------------------------------------------------------------------
# Sweep 1 -- Quantum Foam decoherence, across seeds and noise strength
# ------------------------------------------------------------------

n_seeds = 40
foam_seeds = np.arange(n_seeds)
```

```
noise_levels = [4.0, 8.0, 12.0, 16.0]  # A.4 used 8.0

foam_results = {}  # noise_amp -> array of Psi_foam over seeds
for na in noise_levels:
    vals = [spectral_coherence(make_foam_snapshot(s, noise_amp=na)) for s in foam_seeds]
    foam_results[na] = np.array(vals)

print("=" * 70)
print("SWEEP 1: Quantum Foam decoherence (Psi_foam vs Psi_substrate)")
print("=" * 70)
print(f"Psi_substrate (reference, noiseless) = {Psi_substrate:.4f}\n")
for na, vals in foam_Results:items():
    frac_below = np.mean(vals < Psi_substrate)
    print(f"  noise_amp={na:5.1f}  Psi_foam mean={vals.mean():.4f}  "
          f"std={vals.std():.4f}  min={vals.min():.4f}  max={vals.max():.4f}  "
          f"  fraction decohered (< substrate) = {frac_below:.2%}")

# ---------------------------------------------------------------------
# Sweep 2 -- Universe plateau, across seeds and post-processing params
# ---------------------------------------------------------------------

n_qseeds = 20
qbrane_seeds = np.arange(100, 100 + n_qseeds)
closing_iters = [1, 2, 3, 4]     # A.4 used 2
smoothing_sigmas = [0.6, 1.2, 2.0]  # A.4 used 1.2 (smoothing only affects
                            # the displayed field, not Phi_universe,
                            # which is computed from the binary mask
                            # -- included here for transparency)

phi_qbrane_by_seed = []
phi_universe_by_seed_iter = {it: [] for it in closing_iters}

for qs in qbrane_seeds:
    field = make_qbrane_field(qs)
    giant_mask, phi_q = qbrane_giant_cluster(field)
    phi_qbrane_by_seed.append(phi_q)
    for it in closing_iters:
        universe_mask = ndimage.binary_closing(giant_mask, iterations=it, border_value=1)
        phi_universe_by_seed_iter[it].append(universe_mask.mean())

phi_qbrane_by_seed = np.array(phi_qbrane_by_seed)

print("\n" + "=" * 70)
print("SWEEP 2: Universe plateau (Phi_universe vs Phi_qbrane) [border_value=1, corrected]")
```

```
print("=" * 70)
print(f"Phi_qbrane across {n_qseeds} seeds: mean={phi_qbrane_by_seed.mean():.4f}  "
      f"std={phi_qbrane_by_seed.std():.4f}  min={phi_qbrane_by_seed.min():.4f}  "
      f"max={phi_qbrane_by_seed.max():.4f}\n")
for it in closing_iters:
    vals = np.array(phi_universe_by_seed_iter[it])
    delta = vals - phi_qbrane_by_seed
    frac_at_or_above = np.mean(delta >= -1e-9)
    print(f"  closing_iters={it}  Phi_universe mean={vals.mean():.4f}  "
          f"std={vals.std():.4f}  mean delta over Phi_qbrane={delta.mean():+.4f}  "
          f"  fraction at-or-above qbrane = {frac_at_or_above:.2%}")

# ---------------------------------------------------------------------
# Figure
# ---------------------------------------------------------------------

fig, axes = plt.subplots(1, 2, figsize=(13, 5.2))

ax0 = axes[0]
positions = np.arange(len(noise_levels))
box_data = [foam_results[na] for na in noise_levels]
bp = ax0.boxplot(box_data, positions=positions, widths=0.5, patch_artist=True)
for patch in bp['boxes']:
    patch.set_facecolor('#c9963d')
    patch.set_alpha(0.35)
ax0.axhline(Psi_substrate, color='#1a1a2e', linestyle='--', linewidth=2,
            label=f'Psi_substrate = {Psi_substrate:.3f} (reference)')
ax0.set_xticks(positions)
ax0.set_xticklabels([f'{na:.0f}' for na in noise_levels])
ax0.set_xlabel('Noise amplitude')
ax0.set_ylabel('Psi_foam  (spectral coherence)')
ax0.set_title(f'Foam decoherence across {n_seeds} seeds per noise level')
ax0.legend(fontsize=9)
ax0.grid(alpha=0.25)

ax1 = axes[1]
for it in closing_iters:
    vals = np.array(phi_universe_by_seed_iter[it])
    ax1.plot(qbrane_seeds - qbrane_seeds.min(), vals, '-o', markersize=4,
             label=f'closing_iters={it}', alpha=0.85)
ax1.plot(qbrane_seeds - qbrane_seeds.min(), phi_qbrane_by_seed, 'k--',
         linewidth=2, label='Phi_qbrane (raw, pre-closing)')
ax1.set_xlabel('Q-Brane seed index')
ax1.set_ylabel('Fraction of domain')
```

```
ax1.set_title(f'Universe plateau across {n_qseeds} seeds, varying closing (corrected)')
ax1.legend(fontsize=8)
ax1.grid(alpha=0.25)

fig.suptitle('B.14 -- Robustness sweep: A.4 Quantum Foam & Universe stages (corrected)',
fontsize=13)
fig.tight_layout(rect=[0, 0, 1, 0.95])
fig.savefig('/home/claude/sims/B14_foam_universe_robustness_corrected.png', dpi=180,
facecolor='white')
print("\nSaved figure to B14_foam_universe_robustness_corrected.png")
```

C.9 - Emergence Pipeline - Foam → Q-Brane Coupled Construction (B.28)

emergence_pipeline.py

```
"""
emergence_pipeline.py

A formal-analogy simulation of the five-stage progression described in the manuscript's
cosmological framework:

    Null Point  ->  Quasi-Crystalline Substrate  ->  Quantum Foam
            ->  Q-Brane  ->  Universe (Ignition of Physical Reality)

This is NOT a physical model of the actual universe. It is a mathematical illustration chosen
because each stage admits a well-defined, computable notion of "order" or "coherence," and the
five values chain together into a single curve that has the right qualitative shape for the narrative:
zero -> latent order -> decoherence -> a sharp percolation-type transition ("ignition") -> a stable
plateau.

Two different but complementary order parameters are used, because no single classical statistic
is meaningful across all five stages:

  * Spectral coherence (Psi) -- how sharply peaked a field's power spectrum is. Meaningful for
stages with wave-like / quasiperiodic structure (Substrate, Foam).

  * Connective coherence (Phi) -- the fraction of the domain occupied by the single largest
connected region once a field is thresholded. Meaningful for stages defined by the emergence of
large-scale connected structure (Q-Brane, Universe).

Both are normalised to [0, 1] so they can sit on one axis for the summary plot. The Null Point is
defined analytically as the zero of both (no differentiated structure exists prior to it).

"""
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy import ndimage

rng = np.random.default_rng(7)
L = 300
x = np.linspace(-10, 10, L)
X, Y = np.meshgrid(x, x)

# ----------------------------------------------------------------------
# Stage 0 -- Null Point
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Perfect, undifferentiated symmetry. No broken symmetry means no measurable structure exists
# yet -- coherence is undefined/zero by construction, not computed from a field.
# ----------------------------------------------------------------------

C0 = 0.0

# ----------------------------------------------------------------------
# Stage 1 -- Quasi-Crystalline Substrate
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# Superposition of five plane waves at 72-degree increments. Five-fold symmetric wave sets
# cannot tile a plane periodically, so the resulting density field is quasiperiodic: long-range order
# without translational periodicity -- the formal analogue of a quasi-crystalline substrate.
# ----------------------------------------------------------------------

k0 = 2.0
thetas = np.array([j * 2 * np.pi / 5 for j in range(5)])

substrate_field = np.zeros((L, L))
for th in thetas:
    substrate_field += np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y))

def peak_to_mean_ratio(field):
    F = np.fft.fftshift(np.fft.fft2(field))
    P = np.abs(F) ** 2
    cy, cx = np.array(P.shape) // 2
    P[cy - 2:cy + 3, cx - 2:cx + 3] = 0  # zero out the DC peak/its neighborhood
return P.max() / P.mean()
```

```
# ----------------------------------------------------------------------
# Calibrate the [0,1] spectral-coherence scale against two fixed anchors: pure white noise (the
# least-ordered field possible on this grid) at the bottom, and the noiseless quasicrystal substrate
# (the most-ordered field used in this pipeline) at the top. Everything else is placed on a log scale
# between them, since peak-to-mean ratios span several orders of magnitude.
# ----------------------------------------------------------------------

_noise_floor_ratio = peak_to_mean_ratio(rng.normal(size=(L, L)))
_substrate_ratio = peak_to_mean_ratio(substrate_field)
_log_lo, _log_hi = np.log(_noise_floor_ratio), np.log(_substrate_ratio)
def spectral_coherence(field):
    ratio = peak_to_mean_ratio(field)
    return float(np.clip((np.log(ratio) - _log_lo) / (_log_hi - _log_lo), 0, 1))

Psi_substrate = spectral_coherence(substrate_field)

# ----------------------------------------------------------------------
# Stage 2 -- Quantum Foam
# ----------------------------------------------------------------------

# ----------------------------------------------------------------------
# The same five-wave construction, but now each realization gets random phases, random
# amplitude jitter, and additive white noise -- a seething, fluctuating field rather than a static one.
# Averaging the *power spectrum* (not the field) over many independent realizations shows the
# same underlying wavevectors, but broadened and buried in noise: order is present in principle
# but not resolved in any single instant. This is what should decohere the substrate's sharp peaks.
# ----------------------------------------------------------------------

phases = rng.uniform(0, 2 * np.pi, size=5)
amps = 1 + 0.35 * rng.normal(size=5)
foam_signal = np.zeros((L, L))
for th, ph, a in zip(thetas, phases, amps):
foam_signal += a * np.cos(k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)

# ----------------------------------------------------------------------
# Coherence is measured on a single instantaneous realization, not an ensemble average --
# averaging over many noisy snapshots would recover the underlying wavevectors and defeat the
# point. At any given "moment," strong stochastic noise buries the substrate's order beneath
# foam-like fluctuations, which is what should knock the coherence index down from the
# substrate stage.
# ----------------------------------------------------------------------

foam_snapshot = foam_signal + 8.0 * rng.normal(size=(L, L))
Psi_foam = spectral_coherence(foam_snapshot)
```

```
# ---------------------------------------------------------------------
# Stage 3 -- Q-Brane
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# Many plane waves (16) at random directions and random phases, superposed and thresholded.
# As the threshold is swept, the occupied region undergoes a percolation transition: below
# threshold, occupied cells form small disconnected islands; above a critical value, a single
# connected component suddenly spans a large fraction of the domain. That sudden jump is the
# formal analogue of "ignition" -- a coherent, connected q-brane condensing out of an incoherent
# field.
# ---------------------------------------------------------------------

n_waves = 16
qbrane_base = np.zeros((L, L))
dirs = rng.uniform(0, 2 * np.pi, n_waves)
phases_q = rng.uniform(0, 2 * np.pi, n_waves)
for th, ph in zip(dirs, phases_q):
    qbrane_base += np.cos(1.3 * k0 * (np.cos(th) * X + np.sin(th) * Y) - ph)
qbrane_base /= np.sqrt(n_waves)

# ---------------------------------------------------------------------
# ALPHA controls how much Q-Brane's field genuinely reads Foam's own realized values, as
# opposed to being built independently and merely following Foam in narrative sequence (the
# B.25 finding). At ALPHA=0 this reduces exactly to the original, Foam-independent
# construction. The blend is variance-preserving (sqrt(1-a^2)/a weighting on two unit-variance
# fields) so the two random-wave-model results already reported elsewhere in this appendix stay
# comparable across ALPHA.
# ---------------------------------------------------------------------

ALPHA = 0.3

def _normalize(field):
    return (field - field.mean()) / field.std()

qbrane_field = (np.sqrt(1 - ALPHA ** 2) * _normalize(qbrane_base)
                + ALPHA * _normalize(foam_snapshot))

thresholds = np.linspace(qbrane_field.min(), qbrane_field.max(), 80)
giant_frac = []
for t in thresholds:
    mask = qbrane_field > t
    lbl, n = ndimage.label(mask)
    if n == 0:
```

```
        giant_frac.append(0.0)
        continue
    sizes = ndimage.sum(mask, lbl, index=range(1, n + 1))
    giant_frac.append(sizes.max() / mask.size)
giant_frac = np.array(giant_frac)

# ---------------------------------------------------------------------
# `thresholds` runs low -> high, so `giant_frac` runs *high -> low* (a low threshold admits
# almost the whole field trivially; that is not a meaningful "structure," just an unselective mask).
# The meaningful direction is a high, selective threshold (sparse, disconnected islands) relaxing
# down toward a lower one. Two points on that sweep matter here:
#   t_onset  -- the sharp jump where a giant cluster first spans >20% of the domain (the "ignition"
# instant itself)
#   t_stable -- a further, more relaxed point (>75%) where the ignited structure has fully
# connected and stopped changing qualitatively -- this is the Q-Brane's realized state, not just its
# onset, so it is the value carried into the coherence curve.
# ---------------------------------------------------------------------

rev_frac = giant_frac[::-1]
rev_thresh = thresholds[::-1]

onset_hits = np.where(rev_frac > 0.20)[0]
onset_idx = onset_hits[0] if len(onset_hits) else int(np.argmax(rev_frac))
t_onset = rev_thresh[onset_idx]

stable_hits = np.where(rev_frac > 0.75)[0]
stable_idx = stable_hits[0] if len(stable_hits) else int(np.argmax(rev_frac))
t_crit = rev_thresh[stable_idx]

mask_crit = qbrane_field > t_crit
lbl_crit, n_crit = ndimage.label(mask_crit)
sizes_crit = ndimage.sum(mask_crit, lbl_crit, index=range(1, n_crit + 1))
giant_label = int(np.argmax(sizes_crit)) + 1
giant_mask = lbl_crit == giant_label

Phi_qbrane = rev_frac[stable_idx]

# ---------------------------------------------------------------------
# **Stage 4 -- Universe (Ignition of Physical Reality)**
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# The percolating giant cluster is morphologically closed (small gaps filled) and lightly
```

```
# smoothed -- representing the connected structure settling into a stable configuration rather than
# a marginal, noisy one. The connective coherence here should sit at or slightly above the q-
# brane value and, importantly, hold steady rather than collapse back down -- a plateau rather
# than a spike.
# ----------------------------------------------------------------------

universe_mask = ndimage.binary_closing(giant_mask, iterations=2)
universe_field = ndimage.gaussian_filter(universe_mask.astype(float), sigma=1.2)
Phi_universe = universe_mask.mean()

# ----------------------------------------------------------------------
# Summary
# ----------------------------------------------------------------------

stages = ["Null\nPoint", "Quasi-Crystalline\nSubstrate", "Quantum\nFoam",
          "Q-Brane", "Universe\n(Ignition)"]
values = [C0, Psi_substrate, Psi_foam, Phi_qbrane, Phi_universe]

print("Stage coherence values:")
for s, v in zip(stages, values):
    print(f"  {s.replace(chr(10), ' '):28s} {v:.4f}")
print(f"\nPercolation ignition onset threshold   = {t_onset:.4f}  (giant cluster crosses 20%)")
print(f"Post-ignition stable threshold         = {t_crit:.4f}  (giant cluster crosses 75%)")
print(f"Q-Brane giant-cluster fraction          = {Phi_qbrane:.4f}")
print(f"Stabilized universe fraction           = {Phi_universe:.4f}")

# ----------------------------------------------------------------------
# Figure
# ----------------------------------------------------------------------

fig = plt.figure(figsize=(15, 9))
gs = fig.add_gridspec(2, 5, height_ratios=[1, 0.9])

extent = [x.min(), x.max(), x.min(), x.max()]
ax0 = fig.add_subplot(gs[0, 0])
ax0.set_facecolor('black')
ax0.plot(0, 0, 'o', color='white', markersize=6)
ax0.set_xlim(-1, 1); ax0.set_ylim(-1, 1)
ax0.set_title("Null Point", fontsize=10)
ax0.set_xticks([]); ax0.set_yticks([])

ax1 = fig.add_subplot(gs[0, 1])
ax1.imshow(substrate_field, extent=extent, cmap='cividis')
ax1.set_title("Quasi-Crystalline\nSubstrate", fontsize=10)
```

```
ax1.set_xticks([]); ax1.set_yticks([])

ax2 = fig.add_subplot(gs[0, 2])
ax2.imshow(foam_snapshot, extent=extent, cmap='magma')
ax2.set_title("Quantum Foam\n(single realization)", fontsize=10)
ax2.set_xticks([]); ax2.set_yticks([])

ax3 = fig.add_subplot(gs[0, 3])
disp = np.zeros((L, L, 3))
disp[..., 2] = 0.15 * (qbrane_field > qbrane_field.min())
disp[giant_mask] = [1.0, 0.75, 0.1]
ax3.imshow(disp, extent=extent)
ax3.set_title(f"Q-Brane\n(giant cluster, t={t_crit:.2f})", fontsize=10)
ax3.set_xticks([]); ax3.set_yticks([])

ax4 = fig.add_subplot(gs[0, 4])
ax4.imshow(universe_field, extent=extent, cmap='inferno')
ax4.set_title("Universe\n(stabilized structure)", fontsize=10)
ax4.set_xticks([]); ax4.set_yticks([])

ax5 = fig.add_subplot(gs[1, :])
xs = np.arange(5)
ax5.plot(xs, values, '-o', color='#c9963d', linewidth=2.5, markersize=9,
        markerfacecolor='#1a1a2e', markeredgecolor='#c9963d', markeredgewidth=2)
ax5.fill_between(xs, values, color='#c9963d', alpha=0.12)
ax5.set_xticks(xs)
ax5.set_xticklabels(stages, fontsize=10)
ax5.set_ylabel("Coherence index  (0 = undifferentiated, 1 = fully coherent)")
ax5.set_ylim(-0.05, 1.05)
ax5.set_title("Structural coherence across the five stages", fontsize=12)
ax5.grid(alpha=0.25)
ax5.axvspan(2.5, 3.5, color='#c9963d', alpha=0.08)
ax5.annotate("percolation-type\nignition", xy=(3, values[3]),
            xytext=(3.15, max(values) * 0.55),
            fontsize=9, ha='left',
            arrowprops=dict(arrowstyle='->', color='gray'))
fig.suptitle("Formal-Analogy Simulation: Null Point \u2192 Substrate \u2192 Foam \u2192 Q-
Brane \u2192 Universe",
            fontsize=13, y=0.99)
fig.tight_layout(rect=[0, 0, 1, 0.96])
fig.savefig("/home/claude/emergence_pipeline.png", dpi=180, facecolor='white')
print("\nSaved figure to emergence_pipeline.png")
```

C.10 - Null Point Floor Stress-Test Figure Generator (B.29)

generate_figure_b29.py

```
"""
Figure B.29 - Null Point Floor: Existence, Instability, and Relational Selection

Self-contained reproduction of all four panels. Two pattern-forming PDE classes are used
throughout (Swift-Hohenberg and Ginzburg-Landau), each represented by a single linear
operator L(k) in Fourier space plus a shared cubic nonlinearity -- the standard gradient-flow form
used throughout the B.29 investigation.

Panel 1 (B.29.3) -- entropy/order trajectory under genuine dynamics
Panel 2 (B.29.6-B.29.7) -- existence of a lower-energy state across r
Panel 3 (B.29.9-B.29.10) -- basin-membership fraction vs floor size
Panel 4 (B.29.10) -- orientation/sign selection vs bias strength

Runtime note: panel 4 in particular is iteration-heavy at full grid resolution (N=128) and can take
significant wall-clock time to reproduce exactly as run. Reduce ITERS_* constants for a quick
sanity check of the code path; use the defaults for a faithful reproduction.

"""
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.rcParams['font.family'] = 'DejaVu Sans'
mpl.rcParams['font.size'] = 9.5
mpl.rcParams['axes.titlesize'] = 10.5
mpl.rcParams['axes.titleweight'] = 'bold'

# ---------------------------------------------------------------------
# Shared grid and operators
# ---------------------------------------------------------------------

N, L = 128, 40.0
DX = L / N
xs = np.linspace(-L / 2, L / 2, N, endpoint=False)
X, Y = np.meshgrid(xs, xs)
kx_ax = np.fft.fftfreq(N, d=DX) * 2 * np.pi
KX, KY = np.meshgrid(kx_ax, kx_ax)
K2 = KX**2 + KY**2

LIN_SH = (K2 - 1.0) ** 2   # Swift-Hohenberg: preferred wavenumber |k|=1
D_GL = 1.0                 # Ginzburg-Landau: most unstable mode at k=0
```

```
def L_op_SH(r):
    return r - LIN_SH

def L_op_GL(r):
    return r - D_GL * K2

def step(u, L_op, dt=0.1):

"""One semi-implicit spectral step: linear term implicit (stable for any dt), cubic nonlinearity
explicit."""

    u_hat = np.fft.fft2(u)
    nonlin_hat = np.fft.fft2(u ** 3)
    u_hat = (u_hat - dt * nonlin_hat) / (1 - dt * L_op)
    return np.real(np.fft.ifft2(u_hat))

def relax(u0, L_op, iters, dt=0.1):
    u = u0.copy()
    for _ in range(iters):
        u = step(u, L_op, dt)
    return u

def relax_batched(u0_batch, L_op, iters, dt=0.1):

"""Same as relax(), but u0_batch has shape (trials, N, N); fft2 operates on the last two axes by
default, so all trials run at once."""

    u = u0_batch.copy()
    for _ in range(iters):
        u = step(u, L_op, dt)
    return u

def energy(u, r):

"""Energy functional whose gradient flow is the Swift-Hohenberg dynamics used in panel 2.
E[0] = 0 identically, for any r."""

    local = np.sum(-0.5 * r * u**2 + 0.25 * u**4)
    u_hat = np.fft.fft2(u)
    grad_term = 0.5 * np.sum(LIN_SH * np.abs(u_hat) ** 2) / (N * N)
    return float(local + grad_term)

def fixed_bin_entropy(field, bins=64, value_range=(-2.0, 2.0)):
```

```
"""Coarse-grained Shannon entropy with a FIXED bin range across all frames -- necessary so
entropy growth reflects the dynamics rather than an adaptive-binning artifact (see B.29.3's
discussion of the early ln(2) jump)."""

    hist, _ = np.histogram(field.ravel(), bins=bins, range=value_range)
    p = hist / hist.sum()
    p = p[p > 0]
    return float(-np.sum(p * np.log(p)))

def order_amplitude(field):
    return float(np.sqrt(np.mean(field ** 2)))

def dominant_orientation(field):

"""FFT-based dominant orientation, folded to [0, 180) degrees."""

    F = np.fft.fftshift(np.fft.fft2(field))
    power = np.abs(F) ** 2
    cy, cx = N // 2, N // 2
    yy, xx = np.indices(power.shape)
    r_px = np.sqrt((xx - cx) ** 2 + (yy - cy) ** 2)
    theta = np.arctan2(yy - cy, xx - cx)
    ring = (r_px > 3) & (r_px < 12)
    if power[ring].sum() == 0:
        return None
    ang = theta[ring][np.argmax(power[ring])]
    return np.degrees(ang) % 180
def angular_distance(a, b):
    d = abs(a - b) % 180
    return min(d, 180 - d)

# ----------------------------------------------------------------------
# Panel 1 (B.29.3): entropy + order trajectory under genuine dynamics
# ----------------------------------------------------------------------

def build_panel1():
    sigma_pulse = 1.2
    seed_amp = 0.05
    u0 = seed_amp * np.exp(-(X**2 + Y**2) / (2 * sigma_pulse**2))
    r_val = 0.3
    steps = 300
    dt = 0.05

    S_traj = np.empty(steps + 1)
```

```
    order_traj = np.empty(steps + 1)
    u = u0.copy()
    S_traj[0] = fixed_bin_entropy(u)
    order_traj[0] = order_amplitude(u)
    L_op = L_op_SH(r_val)
    for n in range(1, steps + 1):
        u = step(u, L_op, dt=dt)
        S_traj[n] = fixed_bin_entropy(u)
        order_traj[n] = order_amplitude(u)
    return S_traj, order_traj, dt

# ---------------------------------------------------------------------
# Panel 2 (B.29.6-B.29.7): existence of a lower-energy state across r
# ---------------------------------------------------------------------

def build_panel2(seed=11):
    rng = np.random.default_rng(seed)
    floor = 5e-3
    r_values = np.array([-0.1, -0.05, -0.01, 0.0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.3])
    E_minimized = np.empty(len(r_values))
    for i, r in enumerate(r_values):
        u0 = rng.normal(0, floor, (N, N))
        iters = 3000 if r <= 0.02 else 2000
        u_star = relax(u0, L_op_SH(r), iters=iters, dt=0.1)
        E_minimized[i] = energy(u_star, r)
    return r_values, E_minimized

# ---------------------------------------------------------------------
# Panel 3 (B.29.9-B.29.10): basin-membership fraction vs floor size
# ---------------------------------------------------------------------

def build_panel3(seed=303, r_val=0.05, trials=6):
    rng = np.random.default_rng(seed)
    floor_norms = [1e-1, 1e-2, 1e-3, 1e-4]
    frac_null_SH, frac_null_GL = [], []
    Lsh, Lgl = L_op_SH(r_val), L_op_GL(r_val)

    for norm in floor_norms:
        iters = int(min(9000, 1500 + 3000 * max(0, -np.log10(norm))))
        directions = rng.normal(0, 1, (trials, N, N))
        directions /= np.sqrt(np.mean(directions ** 2, axis=(1, 2), keepdims=True))
        u0_batch = norm * directions

        u_sh = relax_batched(u0_batch, Lsh, iters=iters)
```

```
        rms_sh = np.sqrt(np.mean(u_sh ** 2, axis=(1, 2)))
        frac_null_SH.append(float(np.mean(rms_sh < 1e-3)))

        u_gl = relax_batched(u0_batch, Lgl, iters=iters)
        rms_gl = np.sqrt(np.mean(u_gl ** 2, axis=(1, 2)))
        frac_null_GL.append(float(np.mean(rms_gl < 1e-3)))

        print(f"  panel3: floor={norm:.0e}  frac_null_SH={frac_null_SH[-1]:.2f}  "
              f"frac_null_GL={frac_null_GL[-1]:.2f}")

    return floor_norms, frac_null_SH, frac_null_GL

# ---------------------------------------------------------------------
# Panel 4 (B.29.10): orientation/sign selection vs bias strength
# ---------------------------------------------------------------------

def build_panel4_sh(seed=707, r_val=0.05, trials=6, iters=6000):
    rng = np.random.default_rng(seed)
    noise_level = 1e-3
    bias_angle_deg = 18.4
    bias_kx, bias_ky = 0.9425, 0.3142   # nearest grid-allowed mode to |k|=1
    bias_wave = np.cos(bias_kx * X + bias_ky * Y)
    L_op = L_op_SH(r_val)

    ratios = [0.0, 0.2, 0.5]
    match_fracs = []
    for ratio in ratios:
        bias_amp = ratio * noise_level
        noise = rng.normal(0, noise_level, (trials, N, N))
        u0_batch = noise + bias_amp * bias_wave
        u_star = relax_batched(u0_batch, L_op, iters=iters)
        matches = 0
        for i in range(trials):
            ang = dominant_orientation(u_star[i])
            if ang is not None and angular_distance(ang, bias_angle_deg) < 10:
                matches += 1
        match_fracs.append(matches / trials)
        print(f"  panel4 SH: bias/noise={ratio:.1f}  match_frac={match_fracs[-1]:.2f}")
    return ratios, match_fracs

def build_panel4_gl(seed=808, r_val=0.05, trials=6, iters=3000):
    rng = np.random.default_rng(seed)
    noise_level = 1e-2
    L_op = L_op_GL(r_val)
```

```
    ratios = [0.0, 0.2, 0.5, 1.0]
    match_fracs = []
    for ratio in ratios:
        bias_amp = ratio * noise_level
        noise = rng.normal(0, noise_level, (trials, N, N))
        u0_batch = noise + bias_amp
        u_star = relax_batched(u0_batch, L_op, iters=iters)
        means = np.mean(u_star, axis=(1, 2))
        match_fracs.append(float(np.mean(means > 0)))
        print(f"  panel4 GL: bias/noise={ratio:.1f}  match_frac={match_fracs[-1]:.2f}")
    return ratios, match_fracs

# ----------------------------------------------------------------------
# Assemble the composite figure
# ----------------------------------------------------------------------

def main():
    print("Building panel 1 (entropy trajectory)...")
    S_traj, order_traj, dt = build_panel1()
    print("Building panel 2 (existence across r)...")
    r_values, E_minimized = build_panel2()

    print("Building panel 3 (basin membership vs floor size)...")
    floor_norms, frac_null_SH, frac_null_GL = build_panel3()
    print("Building panel 4 (selection vs bias) -- Swift-Hohenberg...")
    bias_ratios_sh, match_sh = build_panel4_sh()

    print("Building panel 4 (selection vs bias) -- Ginzburg-Landau...")
    bias_ratios_gl, match_gl = build_panel4_gl()

    fig, axes = plt.subplots(2, 2, figsize=(11, 8.5))

# Panel 1

    ax = axes[0, 0]
    t = np.arange(len(S_traj)) * dt
    ax2 = ax.twinx()
    l1, = ax.plot(t, S_traj, color='#1f5fa8', lw=1.8, label='Entropy S(t)')
    l2, = ax2.plot(t, order_traj, color='#c1502e', lw=1.8, ls='--', label='Order (RMS)')
    ax.axvspan(0, t[20], color='grey', alpha=0.15)
    ax.text(t[20] * 0.5, max(S_traj) * 0.92, 'binning-\nresolution\nartifact',
            fontsize=7.5, ha='center', color='dimgrey', style='italic')
    ax.set_xlabel('simulated time (dynamics only - not physical duration of null point)')
    ax.set_ylabel('entropy S(t)', color='#1f5fa8')
```

```
    ax2.set_ylabel('order parameter (RMS)', color='#c1502e')
    ax.set_title('B.29.3 - Nucleation-and-growth entropy curve')
    ax.legend(handles=[l1, l2], loc='upper left', fontsize=8, frameon=False)
```

 # Panel 2

```
    ax = axes[0, 1]
    colors = ['#1f5fa8' if r <= 0 else '#c1502e' for r in r_values]
    ax.axhline(0, color='grey', lw=1, ls=':')
    ax.axvline(0, color='grey', lw=1, ls=':')
    ax.scatter(r_values, E_minimized, c=colors, s=55, zorder=3, edgecolor='k', linewidth=0.4)
    ax.plot(r_values, E_minimized, color='grey', lw=1, alpha=0.6, zorder=2)
    ax.set_yscale('symlog', linthresh=1e-8)
    ax.set_xlabel('r  (instability parameter)')
    ax.set_ylabel('minimized energy E, relative to E[null]=0')
    ax.set_title('B.29.6-B.29.7 - Existence of a lower-energy state')
    ax.text(0.98, 0.05, 'blue: r≤0, null state remains the minimum\norange: r>0, lower-energy state
exists',
            transform=ax.transAxes, ha='right', va='bottom', fontsize=7.5, color='dimgrey')
```

 # Panel 3

```
    ax = axes[1, 0]
    x = np.arange(len(floor_norms))
    width = 0.35
    ax.bar(x - width / 2, frac_null_SH, width, label='Swift-Hohenberg', color='#1f5fa8')
    ax.bar(x + width / 2, frac_null_GL, width, label='Ginzburg-Landau', color='#c1502e')
    ax.set_xticks(x)
    ax.set_xticklabels([f'{n:.0e}' for n in floor_norms])
    ax.set_ylim(-0.05, 1.05)
    ax.set_xlabel('floor magnitude (norm)')
    ax.set_ylabel('fraction landing in null-state basin')
    ax.set_title('B.29.9-B.29.10 - Basin membership vs floor size (r=0.05)')
    ax.legend(fontsize=8, frameon=False)
```

 # Panel 4

```
    ax = axes[1, 1]
    ax.plot(bias_ratios_sh, match_sh, 'o-', color='#1f5fa8', lw=1.8, ms=7,
            label='SH: orientation match')
    ax.plot(bias_ratios_gl, match_gl, 's-', color='#c1502e', lw=1.8, ms=7,
            label='GL: sign match')
    ax.axhline(0.5, color='grey', lw=1, ls=':')
    ax.text(1.02, 0.52, 'chance level\n(no bias)', fontsize=7.5, color='dimgrey', va='bottom')
```

```
    ax.set_xlabel('floor bias amplitude / background noise amplitude')
    ax.set_ylabel('fraction selecting the biased outcome')
    ax.set_ylim(-0.05, 1.15)
    ax.set_title('B.29.10 - Selection sharpness: "which," not just "whether"')
    ax.legend(fontsize=8, frameon=False, loc='lower right')

    fig.suptitle('Figure B.29 - Null Point Floor: Existence, Instability, and Relational Selection',
                 fontsize=12.5, fontweight='bold', y=0.985)
    fig.tight_layout(rect=[0, 0, 1, 0.97])
    fig.savefig('B29_null_point_floor.png', dpi=200, facecolor='white')
    print("Saved B29_null_point_floor.png")

if __name__ == '__main__':
    main()
```

C.11 - Complex-Valued Extensions and Closure Comparison (B.33)

b33_complex_extensions.py

```
"""
C.11 - Complex-Valued Extensions and Closure Comparison (B.33)

Two independent tests, run in support of B.33's discussion of complex-valued
extensions to the B.29 toy models. Neither test bears on which of B.33's
three closures is correct -- both are checks on what complex-number
formulations add to the description of the transition, not on selection
among outcomes.

Test 1 (Panels 1-3): resonant-triad complex amplitude equations. Standard
Ginzburg-Landau/Newell-Whitehead-Segel form for three modes whose
wavevectors satisfy k1+k2+k3=0. Demonstrates phase-locking and discrete
orientation-selection -- phenomena a real-only field cannot represent,
since a real field has no phase to lock or select among.

Test 2 (Panel 4): whether an imaginary diffusion coefficient turns a
relaxational (gradient-flow) fluctuation into a propagating wave. Confirms
the analytic prediction (rotation frequency omega = b*k^2) against direct
numerical integration.
"""

import numpy as np
import json
from scipy.integrate import solve_ivp
```

```
rng_master = np.random.default_rng(7)

# ---------------------------------------------------------------------
# Shared triad dynamics
# ---------------------------------------------------------------------

# ---------------------------------------------------------------------
# dA_i/dt = A_i*(mu - |A_i|^2 - g*sum(other |A_j|^2)) + c*conj(A_j)*conj(A_k)
# The quadratic coupling term is written in the *conjugate* amplitudes,
# which is what k1+k2+k3=0 resonance requires -- this term has no real-only
# analogue with the same structure, since conj(A) = A for a real field.
# ---------------------------------------------------------------------

G_COUPLING = 1.0
C_COUPLING = 0.6

def triad_rhs(mu):
  def f(t, y):
    A = y[0:3] + 1j * y[3:6]
    dA = np.zeros(3, dtype=complex)
    idx = [(0, 1, 2), (1, 0, 2), (2, 0, 1)]
    for i, j, k in idx:
      dA[i] = (A[i] * (mu - abs(A[i]) ** 2 - G_COUPLING * (abs(A[j]) ** 2 + abs(A[k]) ** 2))
          + C_COUPLING * np.conj(A[j]) * np.conj(A[k]))
    return np.concatenate([dA.real, dA.imag])
  return f

# ---------------------------------------------------------------------
# Panel 1 (mu>0): does the field grow from near-zero noise, and does the
# phase sum theta1+theta2+theta3 lock to a fixed value? Two hundred
# independent noise seeds, near-zero initial condition (Null Point analogue).
# ---------------------------------------------------------------------

def panel1_growth_and_locking(n_trials=200, mu=0.05, seed=7):
  rng = np.random.default_rng(seed)
  amps, phase_sums = [], []
  for _ in range(n_trials):
    y0 = rng.normal(scale=1e-3, size=6)
    sol = solve_ivp(triad_rhs(mu), [0, 400], y0, method="RK45",
          max_step=0.5, rtol=1e-8, atol=1e-10)
    A_final = sol.y[0:3, -1] + 1j * sol.y[3:6, -1]
    amps.append(np.abs(A_final).mean())
    phase_sums.append((np.sum(np.angle(A_final)) + np.pi) % (2 * np.pi) - np.pi)
  amps, phase_sums = np.array(amps), np.array(phase_sums)
```

```
  locked = np.minimum(np.abs(phase_sums), np.abs(np.abs(phase_sums) - np.pi)) < 0.05
  return {
    "n_trials": n_trials,
    "mean_final_amplitude": float(amps.mean()),
    "std_final_amplitude": float(amps.std()),
    "fraction_grew_past_floor": float((amps > 0.1).mean()),
    "phase_sum_mean": float(phase_sums.mean()),
    "phase_sum_std": float(phase_sums.std()),
    "fraction_phase_locked": float(locked.mean()),
  }

# ---------------------------------------------------------------------
# Panel 2: r<=0 control (should not grow past the floor) and a real-only
# comparison run on the same equations with phase forced to zero, to show
# what "no imaginary part" looks like on this exact system.
# ---------------------------------------------------------------------

def panel2_controls(n_trials=100, seed=11):
  rng = np.random.default_rng(seed)
  grew = 0
  for _ in range(n_trials):
    y0 = rng.normal(scale=1e-3, size=6)
    sol = solve_ivp(triad_rhs(-0.02), [0, 400], y0, method="RK45",
          max_step=0.5, rtol=1e-8, atol=1e-10)
    A_final = sol.y[0:3, -1] + 1j * sol.y[3:6, -1]
    if np.abs(A_final).mean() > 0.1:
      grew += 1

  def rhs_real(t, y):
    mu = 0.05
    dA = np.zeros(3)
    idx = [(0, 1, 2), (1, 0, 2), (2, 0, 1)]
    for i, j, k in idx:
      dA[i] = y[i] * (mu - y[i] ** 2 - G_COUPLING * (y[j] ** 2 + y[k] ** 2)) + C_COUPLING *
y[j] * y[k]
    return dA

  sol_r = solve_ivp(rhs_real, [0, 400], np.array([1e-3, 1e-3, 1e-3]), method="RK45",
          max_step=0.5, rtol=1e-8, atol=1e-10)
  return {
    "n_trials": n_trials,
    "mu_le_0_fraction_grew": grew / n_trials,
    "real_only_final_amplitudes": sol_r.y[:, -1].tolist(),
```

```
    "real_only_note": "no phase variable exists in this reduction; orientation-selection is not
expressible, only magnitude",
  }

# ---------------------------------------------------------------------
# Panel 3: with zero directional bias, does the system still resolve into
# discrete orientation clusters purely from noise? Three hundred trials,
# tracking the final phase of mode 1.
# ---------------------------------------------------------------------

def panel3_orientation_degeneracy(n_trials=300, mu=0.05, seed=3):
  rng = np.random.default_rng(seed)
  thetas = []
  for _ in range(n_trials):
    y0 = rng.normal(scale=1e-3, size=6)
    sol = solve_ivp(triad_rhs(mu), [0, 400], y0, method="RK45",
          max_step=0.5, rtol=1e-8, atol=1e-10)
    A_final = sol.y[0:3, -1] + 1j * sol.y[3:6, -1]
    thetas.append(np.angle(A_final[0]))
  thetas = np.array(thetas)
  bins = np.round(thetas / (2 * np.pi / 3)) * (2 * np.pi / 3)
  unique, counts = np.unique(np.round(bins, 2), return_counts=True)
  return {
    "n_trials": n_trials,
    "clusters": [{"theta_rad": float(u), "count": int(ct)} for u, ct in zip(unique, counts)],
    "n_distinct_clusters": len(unique),
  }

# ---------------------------------------------------------------------
# Panel 4: does an imaginary diffusion coefficient turn relaxation into a
# traveling wave? Linear reduction of the complex Ginzburg-Landau equation
# for a single Fourier mode: dA_k/dt = A_k*[(mu-k^2) - i*b*k^2]. Analytic
# prediction: rotation frequency omega = b*k^2. Real-field control has no
# phase to rotate, checked via sign-change count (zero possible for
# monotonic decay/growth).
# ---------------------------------------------------------------------

def panel4_wave_test(b=0.8, k=1.5, mu=1.0):
  def rhs(t, y):
    A = y[0] + 1j * y[1]
    dA = A * (mu - k ** 2) - 1j * b * k ** 2 * A
    return [dA.real, dA.imag]

  sol = solve_ivp(rhs, [0, 10], [1e-3, 0.0], method="RK45", max_step=0.01, dense_output=True)
```

```
    t = np.linspace(0, 10, 2000)
    A = sol.sol(t)[0] + 1j * sol.sol(t)[1]
    phase = np.unwrap(np.angle(A))
    measured_omega = float(np.polyfit(t, phase, 1)[0])
    predicted_omega = float(-b * k ** 2)

    def rhs_real(t, y):
      return [y[0] * (mu - k ** 2)]

    sol_r = solve_ivp(rhs_real, [0, 10], [1e-3], method="RK45", max_step=0.01,
dense_output=True)
    u = sol_r.sol(t)[0]
    return {
      "b": b, "k": k, "mu": mu,
      "predicted_omega": predicted_omega,
      "measured_omega": measured_omega,
      "match": bool(np.isclose(measured_omega, predicted_omega, rtol=1e-3)),
      "real_field_final_value": float(u[-1]),
      "real_field_sign_changes": int(np.sum(np.diff(np.sign(u)) != 0)),
    }

def run():
    return {
      "panel1_growth_and_locking": panel1_growth_and_locking(),
      "panel2_controls": panel2_controls(),
      "panel3_orientation_degeneracy": panel3_orientation_degeneracy(),
      "panel4_wave_test": panel4_wave_test(),
    }

if __name__ == "__main__":
    results = run()
    with open("b33_complex_extensions_report.json", "w") as f:
      json.dump(results, f, indent=2)
    print(json.dumps(results, indent=2))


b33_complex_extensions_report.json
{
  "panel1_growth_and_locking": {
    "n_trials": 200,
    "mean_final_amplitude": 0.2632993161855451,
    "std_final_amplitude": 4.0980616289968965e-17,
    "fraction_grew_past_floor": 1.0,
    "phase_sum_mean": -1.3322676295501878e-17,
```

```
    "phase_sum_std": 1.167371926066966e-16,
    "fraction_phase_locked": 1.0
  },
  "panel2_controls": {
    "n_trials": 100,
    "mu_le_0_fraction_grew": 0.0,
    "real_only_final_amplitudes": [
      0.26329931618554503,
      0.26329931618554503,
      0.26329931618554503
    ],
    "real_only_note": "no phase variable exists in this reduction; orientation-selection is not
expressible, only magnitude"
  },
  "panel3_orientation_degeneracy": {
    "n_trials": 300,
    "clusters": [
      {
        "theta_rad": -2.09,
        "count": 92
      },
      {
        "theta_rad": 0.0,
        "count": 106
      },
      {
        "theta_rad": 2.09,
        "count": 102
      }
    ],
    "n_distinct_clusters": 3
  },
  "panel4_wave_test": {
    "b": 0.8,
    "k": 1.5,
    "mu": 1.0,
    "predicted_omega": -1.8,
    "measured_omega": -1.8000000000014118,
    "match": true,
    "real_field_final_value": 3.726653172082715e-09,
    "real_field_sign_changes": 0
  }
}
```

ISBN: 978-0-9685179-7-0